Computational Methods for Analyzing Chemical Graphs and Biological Networks(Dissertation_全文)

AUTHOR(S):

Zhao, Yang

Computational Methods for Analyzing Chemical Graphs and Biological Networks

化学グラフと生体ネットワークに対する情報解析手法

Yang Zhao

趙 楊

Abstract

Complex biological systems, which can be represented as computable networks including transcriptional, signalling and metabolic, have been enormously explored over past few decades. However, few of them are known on their complete structures, physiological functions or organic mechanisms. Therefore, comprehensive study of these types of biological networks with their topological structures and prediction of the complete state description of their organisms are important and methodologically-needed in system biology, which can help to deep understand the biological function, and even to design new drug targets to treat diseases.

In this research, we focus on the analysis of chemical graphs and biological network structures, and propose novel methods that handle four types of problems: (i) tree structured data compression, (ii) tree-like compound enumeration, (iii) protein complex prediction, and (iv) modeling the robustness of metabolic networks.

In the first part of this thesis, our research starts with the analysis of tree structured data, since tree graphs have simple structures with no cycle and no selfloop. First at all, we firstly propose integer programming-based methods for grammar-based tree compression. The results show that our methods can not only efficiently compress trees but also extract features and patterns from glycans. Next, we develop two efficient methods for enumerating tree-like compounds with and without multiple bonds by firstly using breadth first search order. To reduce the exponentially increasing search space with the increasing atoms, we employed and modified some important concepts such as left-heavy, center-rooted and normal form. The results of comparison with state-of-the-art methods indicate that our methods are exact and faster. Furthermore, the low matching rates of generated structures in the database suggest that a large amount of structures are exponentially generated, but only a small fraction of them has been explored. This finding is encouraging in a sense that these unexplored regions may provide a vast potentiality to improve our today's drugs by new compound design.

In the second part, we deal with prediction of protein complexes from large-scale protein-protein interaction networks. We propose an improved integer programming-based method based on the idea that a candidate complex should not be divided into many small complexes. We further enhance it by combining with maximal

components and extreme sets in graph theory. The results suggest that our methods outperform the state-of-the-art one. We prove our proposed prediction problem is NP-hard, which justifies the utilization of integer programming.

The final part of this thesis focuses on the analysis of metabolic networks. Metabolic networks are complex systems of chemical reactions taking place in every living cell to degrade substrates and synthesize molecules needed for life. Modeling the robustness of these networks with respect to the dysfunction of some reactions are important to understand the basic principles of biological network organization, and to identify new drug targets. In this part, we propose a new model, the flux balance impact degree, to model the robustness of large metabolic networks with respect to gene knock-out. We formulate the computation of the impact of one or several reaction blocking as linear programs, and propose efficient strategies to solve them. We show that the proposed method better predicts the phenotypic impact of a single gene deletion on Escherichia coli than existing methods.

Acknowledgments

First and foremost, I would like to express my deepest thanks to my advisor Professor Tatsuya Akutsu, who has supervised this research during my doctoral course. He patiently provided invaluable advice, constructive comments and warm encouragement for me through the doctoral program and writing of this thesis. His immense knowledge and supportive advice not only helped me to complete the dissertation but also guided me to gain many valuable experience and learn many research skills.

I shall extend my thanks to Professor Hiroshi Nagamochi for his valuable comments and helpful suggestions on enumeration of molecules.

I am deeply grateful to Professor Jean-Philippe Vert who advised the study of robustness analysis of metabolic networks, for his great idea, insightful comments and suggestions.

My heartfelt thanks also goes to Assistant Professor Morihiro Hayashida for giving me the chance to gain more valuable experience in study and pointing in the direction of new possibilities for this research. I am also grateful to Assistant Professor Takeyuki Tamura for his helpful advice, technological support, discussion and comments during the study for robustness analysis of metabolic networks.

I would like to particularly thank the members in Akutsu Laboratory for their kindness, encouragement and help in my studies and life in Japan. Acknowledgement also goes to the Bioinformatics Center, Institute for Chemical Research, Kyoto University for providing the computational resources.

The work in Chapter 2 was partially supported by the Grants-in-Aid No. 22240009 and 21700323 from the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT). A part of the work in Chapter 3 was supported by Grants-in-Aid No. 22240009, 24500361, and 25-2920 from MEXT. The work in Chapter 4 was partially supported by the Grant-in-Aid No. 22240009 and 21700323 from MEXT. The work in Chapter 5 was supported in part by Grants-in-Aid No. 22240009, 22650045 and (A) 25250028 from the Japan Society for the Promotion of Science (JSPS), and was also partially supported by the European Research Council (SMAC-ERC-280032). Moreover, I was partially supported by a research fellowship from JSPS for Young Scientists.

Publication Notes

Chapter 2 is based on the paper [97], which is publised in *BMC Bioinformatics*.
Chapter 3 is based on the paper [98], which is published in *Journal of Bioinformatics and Computational Biology*.
Chapter 4 is based on the paper [96], which is published in *International Journal of Bioinformatics Research and Application*.
Chapter 5 is based on the paper [99], which is published in *Bioinformatics*.

Contents

List of Figures **xi**

List of Tables **xvii**

1 Introduction **1**
1.1 Background 1
1.2 Tree-structured biological data 2
1.3 Biological networks 3
1.3.1 Protein-protein interaction networks 3
1.3.2 Metabolic networks 5
1.4 Organization of the thesis 7

2 Integer programming-based methods for grammar-based tree compression **9**
2.1 Background 9
2.2 Grammar-based string compression 10
2.2.1 Minimum CFG problem 10
2.2.2 IP formulation for minimum CFG 11
2.3 Grammar-based ordered tree compression 13
2.3.1 Minimum EOTG problem 13
2.3.2 IP formulation for minimum EOTG 13
2.4 Grammar-based unordered tree compression 16
2.4.1 Minimum EUTG problem 16
2.4.2 IP formulation for minimum EUTG 18
2.5 Results 18
2.5.1 Instances for tree compression 19
2.5.2 Artificial tree compression 21
2.5.3 Glycan tree-structure pattern extraction 23
2.6 Discussion 24

3 Enumeration of tree-like chemical compounds 27
3.1 Background . . . 27
3.2 Preliminaries . . . 29
3.2.1 Molecular trees . . . 29
3.2.2 Center-rooted . . . 29
3.2.3 Left-heavy . . . 31
3.2.4 Normal form . . . 31
3.2.5 Family tree . . . 32
3.3 Methods . . . 35
3.3.1 BfsSimEnum for simple tree enumeration . . . 35
3.3.2 BfsMulEnum for multi-tree enumeration . . . 37
3.3.3 Time complexity analysis . . . 39
3.4 Results . . . 39
3.4.1 Comparison with existing methods . . . 39
3.4.2 Extension to multivalent elements . . . 42
3.4.3 Structural matching . . . 43
3.5 Conclusion . . . 44

4 Protein complex prediction via improved verification methods 47
4.1 Background . . . 47
4.2 Methods . . . 49
4.2.1 Improved integer programming IPc . . . 49
4.2.2 Maximal components and extreme sets . . . 50
4.3 Hardness analysis . . . 54
4.4 Results . . . 57
4.4.1 Data and implementation . . . 57
4.4.2 Comparison with existing methods . . . 58
4.5 Conclusion . . . 62

5 Modeling the robustness of metabolic networks 65
5.1 Background . . . 65
5.2 Flux balance impact degree (FBID) . . . 68
5.3 Methods . . . 69
5.3.1 Elementary mode (EM)-based computation . . . 69
5.3.2 Linear programming (LP)-based computation . . . 70
5.4 Data . . . 71
5.4.1 The Escherichia coli metabolic networks . . . 71
5.4.2 Phenotypic data . . . 71
5.5 Results . . . 72
5.5.1 Implementation . . . 72
5.5.2 Computation time . . . 73

5.5.3 Phenotypic prediction 74
5.6 Discussion 79

6 Conclusion and future work **81**
6.1 Summary 81
6.2 Future directions 83

Bibliography **85**

List of Figures

1.1 Example of glycan G06867 from KEGG GLYCAN database 3
1.2 Example of protein interactions (A) from Protein Data Bank (PDB ID: NUP62/NUP54) and its graphical representation (B) that represents mutual binding relationship within Protein 1, 2 and 3. 4
1.3 Part of the metabolic pathway of *Escherichia coli* K-12 MG1655 from KEGG database [47]. 5
1.4 Overview of major steps of FBA. In the example network, A, B, C, D and E are given as metabolites, double-headed arrows labeled as r_1 and b_2 represent reversible reactions, and unfilled arrows labeled as r_2, r_3, r_4, b_1, b_3 and b_4 represent irreversible reactions. The objective is to maximize production of $b_3 + b_4$. 6

2.1 Illustration of the minimum CFG for a string "abcabcab". 12
2.2 Production rules of simple EOTG with black circles representing tags. 14
2.3 Example of ordered tree $T_{i,t,h,k}$. 15
2.4 Illustration for bisections of the tagged and untagged ordered trees for the production rules. 16
2.5 Example of an ordered tree that is not well compressed. The left ordered tree cannot be divided into the two right trees. However, this is possible if the left tree is an unordered tree. 17
2.6 Example of the unordered tree $T_{i,t,\mathcal{C}}$. Example of the unordered tree $T_{i,t,\mathcal{C}}$ with a set of children of i, $\mathcal{C} = \{c_1, \cdots, c_k\}$. 17
2.7 Production rules generated by our IP-based method for the minimum SEOTG (A) Derivation of production rules generated by our IP-based method for the minimum SEOTG. (B) Production rules generated by our IP-based method for the minimum SEOTG. 19
2.8 Production rules generated by our IP-based method for the minimum SEUTG (A) Derivation of production rules generated by our IP-based method for the minimum SEUTG. (B) Production rules generated by our IP-based method for the minimum SEUTG. 20

2.9 Trees used in experiments for evaluation of our IP-based methods (A) Trees having only vertices with degree at the most two. (B) Trees having vertices with degree more than two. 21

2.10 Extracted patterns from glycan G03655 The label related with the lower endpoint is attached to each edge. Labels, a, b, and c denote GlcNAc, Man, and P, respectively. 25

2.11 Extracted patterns from glycan G04458 The label related with the lower endpoint is attached to each edge. Labels, a, and b denote Xyl, and Glc, respectively. 25

2.12 Extracted patterns from glycan G04666 The label related with the lower endpoint is attached to each edge. Labels, a, b, c, and d denote LFuc, Gal, Xyl, and Glc, respectively. 26

2.13 Extracted patterns from glycan G05058 The label related with the lower endpoint is attached to each edge. Labels, a, b, c, d, and e denote Glc, Man6Ac, Man, GlcA, and 3-en-eryHexA, respectively. . 26

3.1 Example of transforming 2-oxo-glutarate into a rooted ordered multitree T. This transformed molecular tree has $depth(T) = 4$, $maxpath =$ {HOCCCCCOH} and label set {C, O, H}, where num(C) = 5, num(O) = 5 and num(H) = 6. 30

3.2 Illustration of two kinds of center-rooted trees. The thick lines represent one of the longest paths, and the vertices in circles represent the center. 30

3.3 Illustration of inequality $T(u) >_s T(v)$ and $T(u) >_m T(v)$. Substructures in gray areas represent comparative subtrees rooted at u and v in each subfigure. The label set of these examples is $\Sigma = \{C, O\}$ whose order is $C > O$. (a) $T(u) >_s T(v)$ holds since the root of $T(u)$ has a greater label than that of $T(v)$. (b) Comparison of their corresponding descendants in BFS order shows that $T(u) >_s T(v)$. (c) The special case of $T(u) >_m T(v)$ with $T(u) =_s T(v)$, for which further comparison is needed to check their corresponding edge multiplicity in BFS order. Since the edge multiplicity in dot circles of $T(u)$ is greater than the corresponding location of $T(v)$, $T(u) >_m T(v)$ holds. 32

3.4 Illustration of determining a normal form. Since the definition of normal form is based on the ideas of left heavy and center-rooted, the root is one endpoint of the center, further comparison is needed between subtree $T_r(v)$ and $T_v(r)$, only if r is the root and v is the other endpoint of the center. The given tree is determined as a normal tree because $T_r(v) >_m T_v(r)$. 33

3.5 Illustration of $P(T)$ being left-heavy. Suppose that v_k is the last vertex of T in BFS order, $T(v_j)$ is any tree such that $v_k \in T(v_j)$, and v_h and v_l are any left and right siblings of v_j in T. Since T is left-heavy and $T(v_h) \geq T(v_j) > T(v_l)$ holds, $T(v_h) \geq T(v_j) - v_k \geq T(v_l)$ always holds (the right-side). 34

3.6 Illustration of 4 kinds of center-rooted trees with their longest paths including the deepest rightmost vertex v_k (i.e., last vertex in BFS order). The black vertices denote the deepest rightmost vertices and the other endpoints of the longest paths in these trees. 34

3.7 Illustration for determining the possible positions where a new leaf should be added to a current tree. (a) If the deepest leftmost vertex v_l and the deepest rightmost vertex v_k are in the same subtree, a new leaf can be added to only vertices from $parent(v_k)$ to v_{l-1} (within the brace). (b) If v_l and v_k are included in distinct subtrees, a new leaf can be added to vertices from $parent(v_k)$ to v_k (within the brace). 35

3.8 Illustration for determining the largest possible label for v_{k+1}. The black circles are vertices having left siblings in the path from v_{k+1} to the root v_1. BfsSimEnum separately compares all of the subtrees rooted at such vertices with the subtrees rooted at their left siblings to determine the largest possible label. For instance, since $T(v_{k+1}) \subseteq T(v_k)$ and $T(v_g) \subseteq T(v_{g-1})$, the largest possible label at these comparison steps are l_i and l_j, which are the labels of corresponding vertices (light gray circles) of v_{k+1} in $T(v_k)$ and $T(v_{g-1})$, respectively; while since $T(v_h) \not\subseteq T(v_{h-1})$, no comparison is done for v_h and the largest possible label is l_1. The largest possible label for v_{k+1} is then determined by using l_i, l_j and l_1. 37

3.9 Illustration of BfsSimEnum and BfsMulEnum. Algorithm 1 is processed above the dot line; Algorithm 2 is processed below the dot line. Graphs in gray color are considered as invalid by the algorithms and thus are not stored or proceeded any more. It should be noted that Hydrogen atoms are added as leaves at last. 40

4.1 Example of verification by two IP-based methods, IPo and IPc. (a) Example of a protein interaction network and domain-domain interactions. There are six proteins that contain one or two domains, seven potentially interacting domain pairs, and seven potentially interacting protein pairs, where these protein-protein interactions are not shown. (b) The optimal solution by the IP of [63], IPo. Each solid line denotes a protein-protein interaction. Two protein complexes are generated. (c) The optimal solution by our proposed IP, IPc. A larger protein complex is generated. 51

4.2 Illustration of a cut $\{X, V - X\}$ that determines the local edge-connectivity $\lambda_G(g, h)$ between vertices g and h, where the graph G contains the set of 19 vertices $V = \{a, b, \ldots, s\}$ and the set of edges E, each number in this figure denotes the weight w_G of the edge, and the edges without a number are weighted by 1. For the set $X = \{a, b, e, f, g, l, m, k, q\}$, $\sum_{u \in X, v \in V-X} w_G(u, v) = 6$. Then, X gives one of the minimum (g, h)-cuts in Figure 4.3, and $\lambda_G(g, h) = 6$. 52

4.3 Minimum (g, h)-cuts of the graph G in Figure 4.2. There are four cuts given by the sets including the vertex h, $X_1 = \{h\}$, $X_2 = \{h, n\}$, $X_3 = \{h, n, o, p, r, s\}$, and $X_4 = \{c, d, h, i, j, n, o, p, r, s\}$. 53

4.4 Illustration of maximal components and extreme sets. The maximal components and the extreme sets of the graph G in Figure 4.2. Each colored area corresponds to a maximal component (an extreme set and a maximal component). 54

4.5 Illustration of the reduction from 3-dimensional matching (3DM) to the protein complex verification problem (PCVP) in the case of $n = 26$, where $\mathcal{X} = \{A, B, \ldots, Z\}$, $\mathcal{Y} = \{1, 2, \ldots, 26\}$, and $\mathcal{Z} = \{a, b, \ldots, z\}$. The large and small circles denote proteins and domains, respectively. The solid and dotted lines denote potential domain-domain interactions, and the solid lines are selected. 56

4.6 Results of the precision by IPo, IPc, maximal, extreme, maximal+IPc, and extreme+IPc for candidates obtained from WI-PHI (top) and BioGRID (bottom) by MCL with varying the inflation parameter from 1.5 to 2.5. 'maximal+IPc' and 'extreme+IPc' denote that IPc is applied after the calculation of maximal components and extreme sets, respectively. Each method was applied to candidate protein complexes. 59

4.7 Results of the precision by IPo, IPc, maximal, extreme, maximal+IPc, and extreme+IPc for candidates obtained from WI-PHI (top) and BioGRID (bottom) by MCODE with varying the node score cutoff parameter from 0.0 to 0.3. 60

5.1 The elementary modes of an example network, where A, B, C, D and E (cycles) are given as metabolites which need fulfill a steady-state. Double-headed arrows labeled as r_1 and b_2 represent reversible reactions. Unfilled arrows labeled as r_2, r_3, r_4, b_1, b_3 and b_4 represent irreversible reactions. Elementary modes of this example are given in the table, where each row represents an elementary mode in which value 0 means that the corresponding reactions are not included in this elementary mode. (See also an metatool format of this example in supplementary materials which can be directly used for open software.) 67
5.2 FBID distribution for the 1366 genes of the KEIO collection dataset computed on the iJO1366 metabolic network. 74
5.3 ROC curves for phenotype prediction from the FBID on various datasets, using both the iJE660 metabolic network (top) and the larger iJO1366 network (bottom). 76
5.4 ROC curve (top) and precision-recall curve (bottom) for phenotype prediction with FBID and FBA on the Keio dataset using the iJO1366 network 78
5.5 FBID and FBA scores for all genes in the Keio dataset analyzed with the iJO1366 network. Red circles correspond to experimentally essential genes. 79

List of Tables

2.1 Results on the elapsed time (seconds) for ordered and unordered trees of type A and B Results on the elapsed time (seconds) for ordered and unordered trees of type A and B (in Figure 2.9) with several sizes. m was the same as the minimum number of nonterminal symbols for each tree, except the case denoted by '*'. '−' denotes that the solver took more than 8 hours. 22

2.2 Results of twelve glycans on the grammar size and the elapsed time (seconds) by our proposed IP-based methods for both the minimum SEOTG and SEUTG problems, and TREE-BISECTION [2]. 23

3.1 Comparison of BfsSimEnum with existing methods. 41

3.2 Comparison of BfsMulEnum with existing methods. 42

3.3 Results for compounds with multivalent elements. 43

3.4 Matching results for chemical compounds only with single bonds to PubChem. 44

3.5 Matching results for chemical compounds with multiple bonds to PubChem. 45

4.1 Results of the precision, the recall, and the accuracy by the original IP-based method (IPo), our improved IP-based method (IPc), maximal components, extreme sets, maximal+IPc, and extreme+IPc for candidate protein complexes obtained from the WI-PHI data by MCL with inflation 1.9. 61

4.2 Results of the precision, the recall, and the accuracy by the original IP-based method (IPo), our improved IP-based method (IPc), maximal components, extreme sets, maximal+IPc, and extreme+IPc for candidate protein complexes obtained from the WI-PHI data by MCL with inflation 2.0. 61

4.3 Results of the precision, the recall, and the accuracy by the original IP-based method (IPo), our improved IP-based method (IPc), maximal components, extreme sets, maximal+IPc, and extreme+IPc for candidate protein complexes obtained from the WI-PHI data by MCL with inflation 2.1. 62

5.1 The *E. coli* network with different versions. 71
5.2 Computational time for FBID computation. 73
5.3 Performance of FBID on gene essentiality prediction, using both iJE660 and iJO1366. 75
5.4 Comparison of the accuracy of FBID with different methods using the iJE660 network. 77

Chapter 1

Introduction

1.1 Background

One purpose for bioinformatics analysis is to systematically describe all molecules and their interactions in a living cell. Most complex interactions between living cells' numerous elements such as proteins, DNA, RNA and small molecules have been demonstrated to take important biological characteristics that control the behavior of the cells [34, 51]. These various inter-cellular networks of interactions in living cells construct complex biological systems that include protein-protein interaction, metabolic, signalling and transcriptional networks. Completely understanding the structures and dynamics of these various types of biological networks within living cells is a fundamental framework for interpretation of living organisms and function needed for life.

The rapid development of high-throughput computational techniques provides researchers with a technology platform to collect the whole information of living cells such as how and when their inside molecules interact with each other. Various types of these complex biological systems in many species have been enormously explored over past few decades, which have been collected as large-scale datasets in several biological databases such as KEGG (Kyoto Encyclopedia of Genes and Genomes [47] at Kyoto University in Japan), BioGRID (the Biological General Repository for Interaction Datasets [83] in Canada), and PubChem[1] (an open website for information on the biological activities of small molecules), etc. These biological databases provide references to facilitate bioinformatics analysis of large-scale datasets, and also produce bases for understanding life as complex biological systems and for developing new drug targets, or other practical applications. However, such data collection is far from complete, since few of these datasets collected from living organisms are known on their complete structures, physiological functions or organic mechanisms.

[1] http://pubchem.ncbi.nlm.nih.gov

Therefore, it is important to develop theoretical and experimental methodologies for mapping, understanding and modelling the topological structures and dynamic properties of the various biological networks, which can also lead to deep understanding of biology, and even to design of new drug targets in disease pathologies.

With the rapid advances of developing theory towards complex networks arising in recent years, it has been demonstrated that most complex networks in nature are sharing the similar organizing principles [85]. This finding allows obtainable principles from well-studied non-biological networks to be used for characterizing cells' functional organization from complex biological systems. Nowadays, some network measures, (such as scale-free networks [5], degree distribution and network robustness [10, 12, 22, 90], etc.), have been utilized to network biology to develop theoretical methodologies and models for understanding the topological structures and cellular organization of complex biological systems in living cells. However, these existing approaches have not been widely used to treat real-world problems properly since their computational inefficiency for large-scale datasets or the shortage of feature properties under taken into account.

In this thesis, our goal is to develop novel methods to topological analyze various biological systems by using graph theory and computational techniques, which can help to efficiently characterize large-scale topological structures and organizing cellular function of these networks. In the rest of this chapter, we briefly review some fundamental knowledge of various biological datasets together with their state-of-the-art methodologies which are also related to our work.

1.2 Tree-structured biological data

As is well known, tree graphs have simple structures with no cycle and no selfloop. There are numerous datasets in biological databases that have tree structures such as glycans, tree-like molecules, etc. Such tree structured biological data can be simply handled as (un)rooted (un) ordered trees by coloring their vertices and edges with special biological properties. Figure 1.1 gives an example of glycan G06867 from KEGG GLYCAN database[2], whose vertices represent monosaccharides and edges represent their glycosidical links. One other example is the representation of tree-like molecules. A tree-like compound can be defined as a connected multi-graph with vertices colored by the atomic symbols of the periodic table, edges labeled by chemical bonds, and edge multiplicity labeled by the atom valence [25].

Several approaches have indicated that most of tree structured biological data are carrying important properties, absence of which will cause the cellular dysfunction [39]. Simultaneously, same patterns from distinct tree structured biological data are also supposed to lead to similar properties in their cellular function, study

[2]http://www.genome.jp/dbget-bin/www_bget?gl:G06867

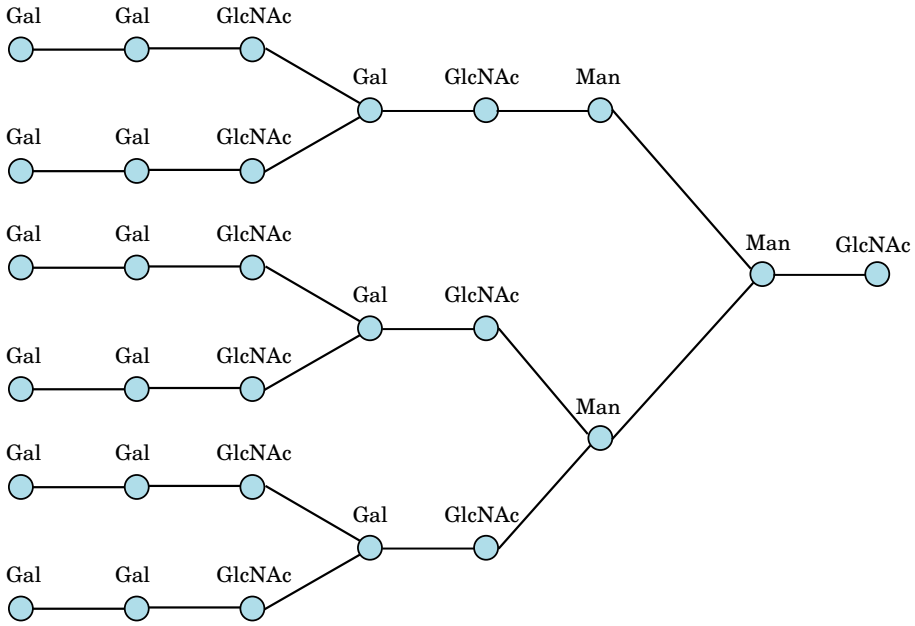


Figure 1.1: Example of glycan G06867 from KEGG GLYCAN database

of which can help to functional molecular design. Thus, it is essential to develop useful and efficient methods for understanding of features or pattern extraction from such datasets.

Our research starts with analysis of tree structured biological data and deals with two types of problems including tree compression and tree-like molecular enumeration. Chapter 2 gives grammar-based methods for compressing tree-structured data, which are the first methods for finding the minimum size grammars for ordered and unordered trees. We also successfully apply them to patten extraction of glycans. In Chapter 3, we propose efficient methods for enumerating tree-like compounds, which is a fundamental framework for molecular design, or even for design of new drug targets to treat diseases. Then, we discuss that our proposed methods are feasible to extend to handle more complex data structures with cycles.

1.3 Biological networks

1.3.1 Protein-protein interaction networks

Protein-protein interaction (PPI) networks are formed by protein-protein interactions in living cells which can be represented as a graph with vertices denoting proteins and edges denoting their interactions (see also an example in Figure 1.2

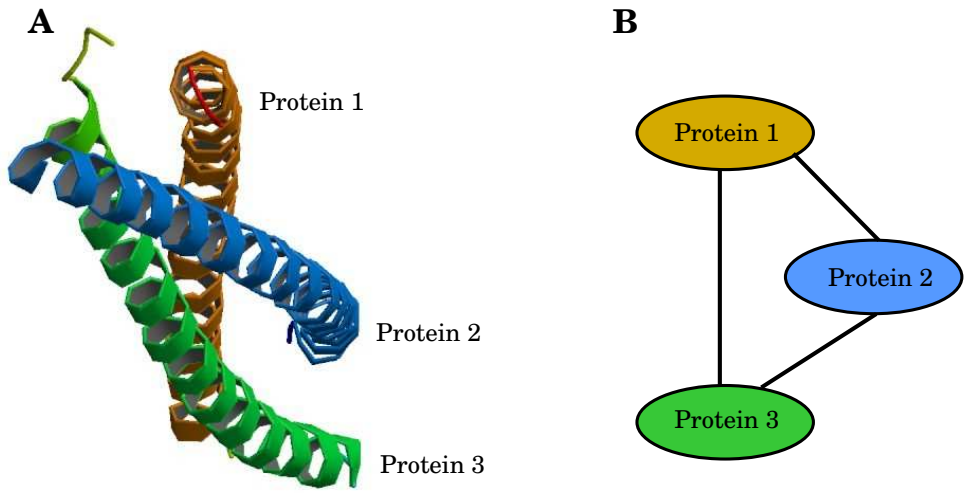


Figure 1.2: Example of protein interactions (A) from Protein Data Bank (PDB ID: NUP62/NUP54) and its graphical representation (B) that represents mutual binding relationship within Protein 1, 2 and 3.

referred from Protein Data Bank[3] with ID: NUP62/NUP54). We can see from Figure 1.2 (B) that PPI networks are represented undirected and explain mutual binding relationship within proteins in such networks.

Protein complexes are known as clusters of multiple proteins linked by non-covalent physical protein-protein interactions that generally correspond to dense regions within PPI networks. Several achievements indicate that protein complexes govern important cellular organization and function of molecular networks [58, 81, 95]. As PPI data grows rapidly, identifying protein complexes within PPI networks becomes necessary and important due to limited availability of known protein complexes. Several methods have been proposed to predict protein complexes from large-scale PPI networks, such as MCL [23], MCODE [8], etc. However, one problem that current methods face is that they detect dense regions as protein complexes without taking into account of structural constraints of proteins, which will lead to numerous false noise. There is a clear need to further verify the candidate complexes which are detected by prediction measurements to facilitate a high precision. Herein, we propose an improved methodology in Chapter 4 for verification of candidate complexes based on the idea of Ozawa et al. [63].

[3]http://www.rcsb.org/pdb/home/home.do

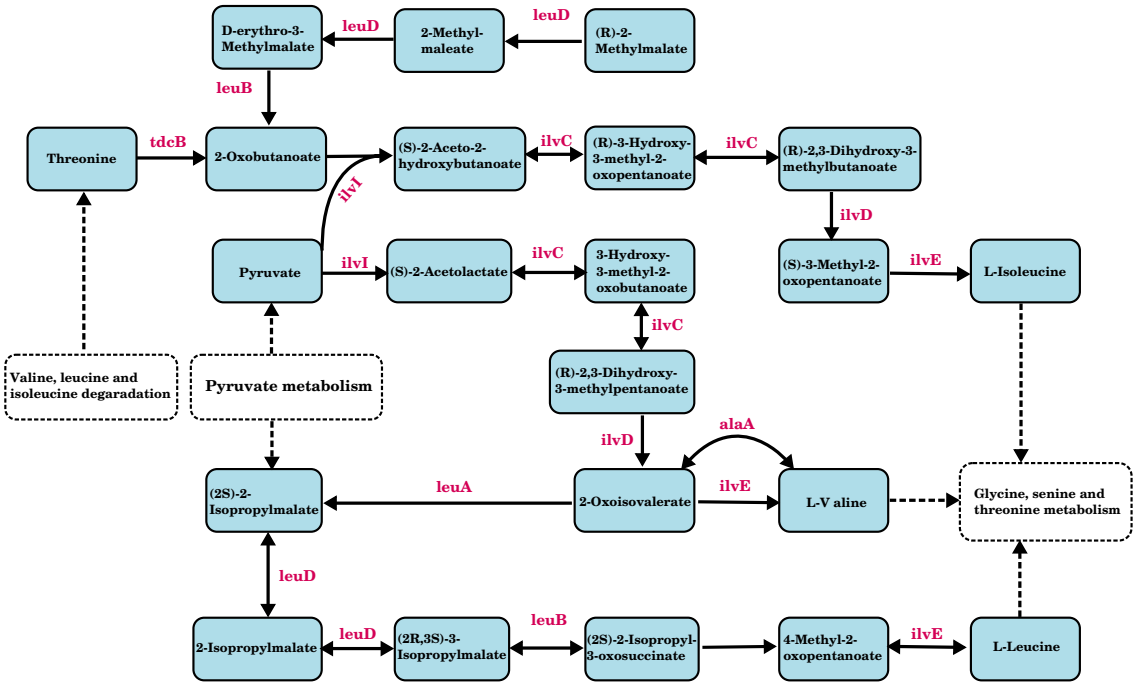


Figure 1.3: Part of the metabolic pathway of *Escherichia coli* K-12 MG1655 from KEGG database [47].

1.3.2 Metabolic networks

Metabolic networks are complex systems of chemical compounds and their enzyme (gene-)catalyzed reactions taking place in every living cell to degrade substrates and synthesize molecules needed for life. Figure 1.3 gives an example of the *Escherichia coli* K-12 MG1655 network from KEGG database[4] [47] From this example, we see that the metabolic network can be represented by a directed graph, whose vertices (blue rectangles) denote reactants or products, edges denote reactions and labels of the edges denote genes that catalyze the reactions. It is also noted that a metabolic network allows internal and exchange reactions (dashed and dot lines in Figure 1.3, respectively) so that the former ones always occur inside the cell, while the latter ones transform metabolites in and out of the cell. In addition, there is no one-to-one correspondence between enzymes (genes) and catalyzed reactions such that an enzyme (gene) can catalyze many different reactions, or conversely one reaction can be catalyzed many different enzymes (genes). The up left of Figure 1.4 shows an example.

[4]http://www.genome.jp/kegg/pathway/eco/eco00290.html

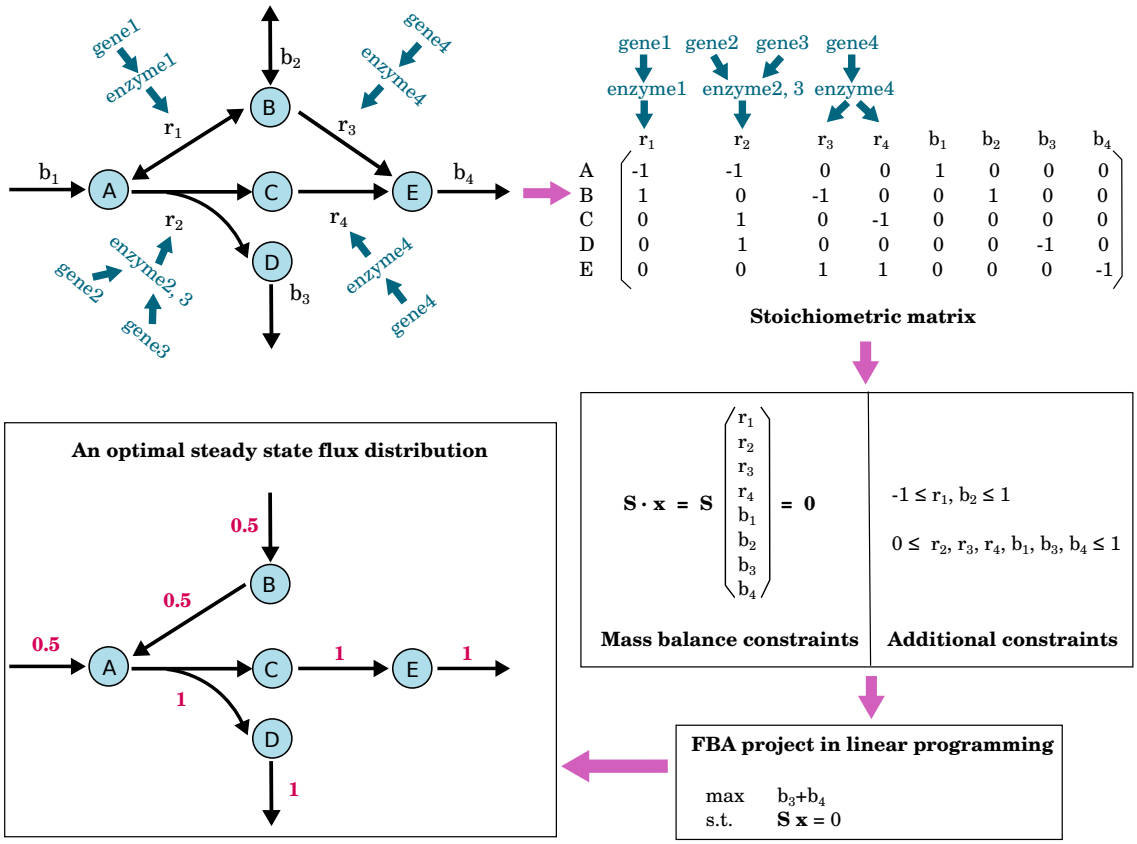


Figure 1.4: Overview of major steps of FBA. In the example network, A, B, C, D and E are given as metabolites, double-headed arrows labeled as r_1 and b_2 represent reversible reactions, and unfilled arrows labeled as r_2, r_3, r_4, b_1, b_3 and b_4 represent irreversible reactions. The objective is to maximize production of $b_3 + b_4$.

Topological network robustness

Robustness of metabolic networks quantifies the ability of metabolites or reactions to respond gene perturbations or mutations while maintaining networks' normal behavior. Understanding and modeling the organizational principles underlying the robustness of metabolic networks with respect to gene perturbations is important not only to shed light on basic principles of life, but also to identify weaknesses which may lead to new drug targets to kill pathogens or cancer cells [12].

Mathematically, we represent a metabolic network as its stoichiometric matrix $\mathbf{S}$ which interprets relative quantities of reactants and products within the network (see an example in Figure 1.4). The dynamic mass balance of such a network in steady state always satisfies $\mathbf{S} \cdot \mathbf{x} = 0$, where $\mathbf{x}$ is a flux vector that represents mass

of substrates being converted into products. Flux balance analysis (FBA), one of the most famous approaches for analyses of the effect of perturbations (occurred by gene deletions or drug inhibitions), uses the mass balance constraint together with a biological relevant objective function to identify the optimal flux distribution [67, 89]. Figure 1.4 gives an overview of major steps of FBA.

Although FBA has been widely used to topologically analyze the robustness of various metabolic networks, it is sometimes difficult to find a compatible objective function. To be independent from the objective function, we propose a new definition of impact degree to model the robustness of large metabolic networks with respect to gene knock-out (Chapter 5).

1.4 Organization of the thesis

The organization of the rest of this thesis is as follows.

In Chapter 2, we propose integer programming-based methods for grammar-based compression of sequence data and tree-structured data. According to some computational experiments, we indicate that our methods can efficiently find the minimum grammars for (un)ordered trees.

In Chapter 3, we deal with enumeration problem of tree-like compounds and propose two efficient algorithms by firstly utilize the breadth first search order. To reduce the large search space, we employ some important concepts such as center-rooted, left-heavy and normal form. We also perform some computational experiments to show our proposed algorithms are exact and efficient. The results show that our proposed algorithms outperform state-of-the-art methods.

In Chapter 4, we propose an improved integer programming-based method for prediction of protein complexes from large-scale protein-protein interaction networks. This approach is based on the idea that a candidate complex should not be divided into many small complexes, and combination methods with maximal components and extreme sets. The results of computational experiments suggest that our methods outperform the existing method. Furthermore, we prove that our proposed verification problems are NP-hard, which justifies the use of integer programming.

In Chapter 5, we propose a new model – the *flux balance impact degree* (FBID) – to model the robustness of large metabolic networks with respect to gene knock-out. We formulate the computation of the impact of one or several reaction blocking as linear programs, and propose efficient strategies to solve them. Finally, we show that our proposed method better predicts the phenotypic impact of single gene deletions on *Escherichia coli* than existing methods.

In Chapter 6, we summarize the conclusion of this thesis, and discuss the future work of presented approaches.

Chapter 2

Integer programming-based methods for grammar-based tree compression

2.1 Background

Traditional data compression (e.g. Huffman coding, arithmetic coding, etc.) has been used to reduce the consumption of expensive resources such as hard disks. Recent achievements successfully apply this technique to analysis of biological data [37, 38, 56]. Among these approaches, Li et al. proposed USM (the universal similarity metric) to approximate the dissimilarity by using compression sizes. They applied a compression algorithm to unaligned mitochondrial genomes, and obtained a phylogeny that was consistent with the commonly accepted one [56]. One other achievement by Hayashida et al. compress the protein tertiary structures and metabolic networks so as to measure their similarities [37, 38].

Grammar-based compression, as a typical data-compression method, aims to seek a small grammar to generate a given string. As is well known, it is NP-hard to find the smallest context-free grammar (CFG) of a given string. Several polynomial time algorithms have recently been proposed to approximate the smallest grammar of input strings within a factor of $O(\log(n/m))$ [15, 70, 71]. Here, n and m are the sizes of the input data and the smallest grammar, respectively. These approaches are able to be used to compress biological sequencing data such as DNA, RNA, and amino acid sequences, etc. Nevertheless, there still exists enormous tree-structured datasets in biological databases such as glycan, etc. Depressedly, few of these achievements are available to treat such kind of datasets. Therefore, it is necessary and important to develop methods to compress tree-structured data.

Recent approaches show that it is feasible to extend the grammar-based compres-

sion to the tree-structure data [14, 59, 93]. However, neither of these methods can output the minimum grammar, nor they can achieve a guaranteed approximation ratio. Simultaneously, Akutsu et al. proposed a bisection algorithm to compress ordered trees by employing existing grammar-based string-compression algorithms [2]. Their method successfully treated given ordered trees by compressing their Euler strings. Although the results of their method have indicated good performance in ordered tree compression, there do not always exist tree grammars corresponding to string grammars derived by the approach.

In this study, our purpose is not only to compress trees but also to deal with pattern extracting from input trees. Herein, we propose an integer programming (IP)-based method that finds the minimum CFG for a given string under the condition that at most two symbols appear on the right-hand side of each production rule. And then, we extend this method to find the minimum elementary ordered-tree grammar (EOTG) and elementary unordered-tree grammar (EUTG) for given ordered and unordered trees. To the best of our knowledge, these are the first methods that can find the minimum size grammars for strings, ordered trees, and unordered trees. Finally, we perform some computational experiments, and also apply our methods to Glycan tree structures for pattern extraction.

2.2 Grammar-based string compression

2.2.1 Minimum CFG problem

We use a simple context-free grammar (CFG) for string and text compression. CFG is defined as 4-tuple $(\Sigma, \Gamma, S, \Delta)$, where Σ is a set of terminal symbols (denoted by a lower-case letter), Γ is a set of nonterminal symbols (denoted by an upper-case letter), S is a start symbol in Γ, and Δ is a set of production rules. The *size* of CFG is defined as the total number of letters appearing on the RHSs (Right-Hand Sides) of production rules. Two CFGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are said to be *equivalent* if $\mathcal{G}_1$ generates the same set of strings as $\mathcal{G}_2$ does. We only consider CFGs consisting of the following types of production rules:

- $A \to a$,
- $A \to BC$.

We call this CFG *a simple CFG*. We can show that any CFG of size m can be transformed into an equivalent, simple CFG of size $3m$.

The smallest grammar problem is thus defined as the problem of finding the smallest grammar that generates a given string [15].

Minimum CFG

Input: String $s = s_1 s_2 \dots s_n$ and integer m.

Output: Simple CFG with m nonterminal symbols that generates s only.

2.2.2 IP formulation for minimum CFG

We use the number of nonterminal symbols m instead of the size of the grammar because the number of terminal symbols appearing in production rules of $A \to a$ is constant for the given string. In order to solve this minimum CFG problem, we propose an IP-based method as follows. We transform this problem into the following integer program, where $x_{1,n} = 1$ holds iff there exists a required CFG $\mathcal{G}$.

Maximize $x_{1,n}$
Subject to

$x_{i,i} = 1$	for all $i = 1, \dots, n$	(1)
$x_{i,j} \leq \sum_{k=i}^{j-1} y_{i,k,j}$	for all $1 \leq i < j \leq n$	(2)
$y_{i,k,j} \leq \frac{1}{2}(x_{i,k} + x_{k+1,j})$	for all $1 \leq i \leq k < j \leq n$	(3)
$z_u \geq \frac{1}{n} \sum_{i,j:s_{i,j}=u} x_{i,j}$	for all distinct substrings u of s	(4)
$\sum_u z_u = m$		(5)

In the above equations, each variable of $x_{i,j}$, $y_{i,k,j}$, and z_u takes either 0 or 1. Each $x_{i,j}$ corresponds to substring $s_{i,j} = s_i s_{i+1} \dots s_j$, and $x_{i,j} = 1$ iff there exists a nonterminal symbol $A_{i,j}$ in $\mathcal{G}$ that generates $s_{i,j}$. $y_{i,k,j} = 1$ iff both of $x_{i,k} = 1$ and $x_{k+1,j} = 1$ hold. It means that $s_{i,j}$ can be generated by concatenating $s_{i,k}$ and $s_{k+1,j}$ that are generated from nonterminal symbols $A_{i,k}$ and $A_{k+1,j}$, respectively. $z_u = 1$ iff there exists a nonterminal symbol in $\mathcal{G}$ that generates a substring u of s. The meaning of each (in)equality is as follows:

(1) each $s_{i,i}$ $(= s_i)$ must be generated,

(2,3) if $A_{i,j}$ appears in $\mathcal{G}$, $s_{i,j}$ must be generated, that is, for at least some k, both of $s_{i,k}$ and $s_{k+1,j}$ must be generated and the production rule $A_{i,j} \to A_{i,k}A_{k+1,j}$ must appear in $\mathcal{G}$,

(4) $A_{i,j}$ and $A_{i',j'}$ are identified if both generate the same substring u, and

(5) the number of nonterminal symbols used in $\mathcal{G}$ must be m.

Figure 2.1 shows an example of the above IP formulation for the string "abcabcab". In the minimum CFG, the substring "abc" is generated from a nonterminal symbol A_{abc}. Then, $z_{abc} = 1$, $x_{1,3} = 1$, and $x_{4,6} = 1$. To generate "abcabc", that is, $x_{1,6} = 1$, at least one variable of $y_{1,k,6}$ $(k = 1, \cdots, 5)$ must be 1, that is, the substring

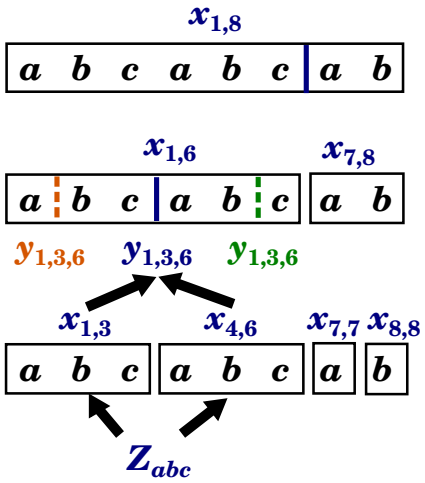


Figure 2.1: Illustration of the minimum CFG for a string "abcabcab".

$s_{1,6}$ must be divided into $s_{1,k}$ and $s_{k+1,6}$. Here, $y_{1,3,6} = 1$ because $x_{1,3} = x_{4,6} = 1$. Then, the production rule $A_{abcabc} \to A_{abc}A_{abc}$ appears.

For this example, the following grammar is constructed from a solution of IP:

$$\begin{aligned} A_{1,8} &\to A_{1,6}A_{7,8}, \\ A_{1,6} &\to A_{1,3}A_{4,6}, \\ A_{1,3} &\to A_{1,2}A_{3,3}, \\ A_{4,6} &\to A_{4,5}A_{6,6}, \\ A_{7,8} &\to A_{7,7}A_{8,8}, \\ &\cdots. \end{aligned}$$

On the other hand, we have $A_{abcabcab} = A_{1,8}$, $A_{abc} = A_{1,3} = A_{4,6}$, etc. Therefore, we finally have:

$$\begin{aligned} A_{abcabcab} &\to A_{abc}A_{abc}A_{ab}, \\ A_{abc} &\to A_{ab}A_{c}, \\ A_{ab} &\to A_{a}A_{b}, \\ A_{a} &\to a, \\ A_{b} &\to b, \\ A_{c} &\to c. \end{aligned}$$

2.3 Grammar-based ordered tree compression

2.3.1 Minimum EOTG problem

We use a simple elementary ordered tree grammar (EOTG) [2] for rooted tree compression. In this grammar, a tree can contain a vertex called *a tag*. A tag indicates that another tree at the root can be attached to it. We assume that there is at most one tag in such a tree.

A simple EOTG (SEOTG) is defined as 4-tuple $(\Sigma, \Gamma, S, \Delta)$, where Σ is a set of terminal symbols, Γ is a set of nonterminal symbols; each edge of the trees has either a terminal or a nonterminal symbol; S is a start symbol in Γ, and Δ is a set of production rules (R1u,t), (R2u,t,t'), and (R3u,t), as in Figure 2.2. (R1u) ((R1t)) denotes a rule when an untagged (tagged) edge of nonterminal symbol A is replaced with an untagged (tagged) edge of terminal symbol a. (R2u,t,t') denotes a rule when an edge of a nonterminal symbol A is replaced with a tree that contains the upper endpoints of edges of nonterminal symbols B and C as the root, and the lower endpoints as two children. (R3u,t) denotes a rule when an edge of A is replaced with a tree in which the root is the upper endpoint of an edge of B, and the lower endpoint is the upper endpoint of an edge of C. We can show that any EOTG of size m can be transformed into an equivalent SEOTG of size $3m$.

Within the class of SEOTGs, we can transform the minimum grammar problem into the IP. For this purpose, we define the minimum SEOTG problem as follows.

Minimum SEOTG

Input: Rooted ordered tree $T(V, E)$ and integer m, where V is a set of vertices and E is a set of labeled edges.

Output: Simple EOTG with m nonterminal symbols that generates only T.

2.3.2 IP formulation for minimum EOTG

A subtree of $T(V, E)$ can be represented as a rooted ordered tree $T_{i,t,h,k}$ with a root i, a tag t, a left-most child h, and a right-most child k of i (Figure 2.3), where the tree includes all the children of i between h and k in $T(V, E)$. If a subtree does not contain any tag, we introduce ϵ that is not included in V, and represent it as $T_{i,\epsilon,h,k}$. It is obvious from the production rules of simple EOTGs that it is sufficient to consider only such subtrees $T_{i,t,h,k}$ for $T(V, E)$

because (R2u,t) denotes a rule to horizontally divide a tree into two trees at the root, and (R3u,t) denotes a rule to vertically divide a tree into two trees at an internal vertex that becomes a tag. Let $ch(i) = (lch(i), \cdots, rch(i))$ denote a sequence of all the children of i in $T(V, E)$, $lch(i)$ is the left-most child, and $rch(i)$ is the right-most child. Without loss of generality, we assume that $i_1 \leq \cdots \leq i_k$ for

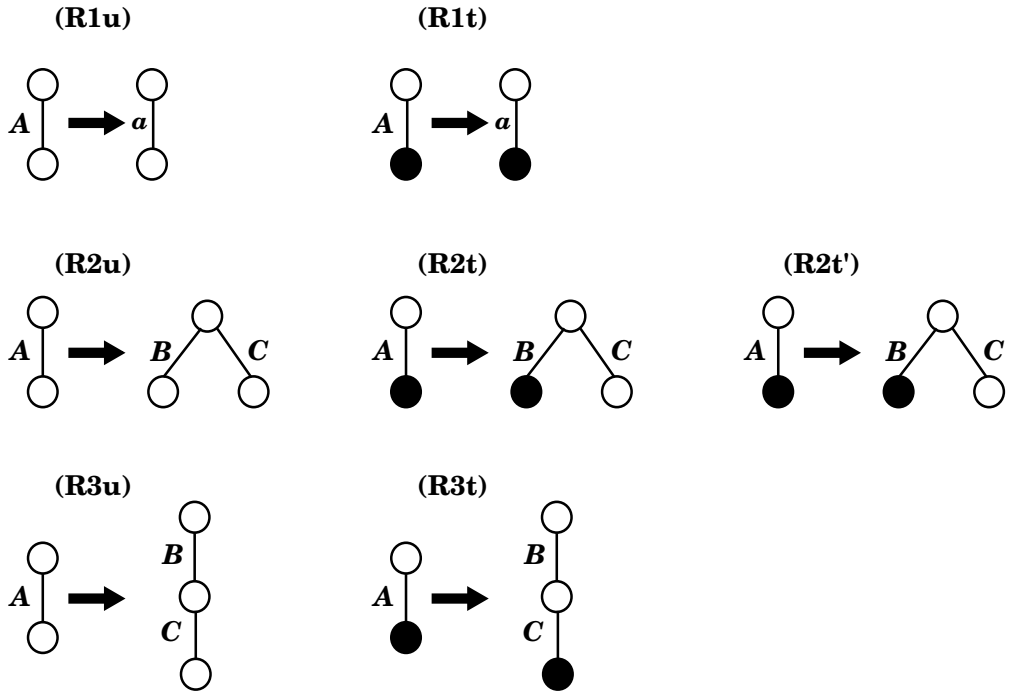


Figure 2.2: Production rules of simple EOTG with black circles representing tags.

$ch(i) = (i_1, \cdots, i_k)$. We suppose that the root of $T(V, E)$ is 1. Then, this problem can be transformed into the following integer program, where $x_{1,\epsilon,lch(1),rch(1)} = 1$ holds iff there exists a required EOTG $\mathcal{G}$.

Maximize $x_{1,\epsilon,lch(1),rch(1)}$
Subject to

$$x_{i,\epsilon,j,j} = 1 \quad \text{for all } i,\ j \in ch(i)\ (|ch(j)| = 0) \tag{1u}$$
$$x_{i,j,j,j} = 1 \quad \text{for all } i,\ j \in ch(i)\ (|ch(j)| > 0) \tag{1t}$$
$$x_{i,\epsilon,h,k} \leq \sum_{l=h}^{k-1} y^{ho}_{i,\epsilon,h,l,k} + \sum_{t \in I(T_{i,\epsilon,h,k})} y^{ve}_{i,\epsilon,h,k,t} \quad \text{for all } i,\ h \leq k \in ch(i) \tag{2-3u}$$
$$y^{ho}_{i,\epsilon,h,l,k} \leq \frac{1}{2}(x_{i,\epsilon,h,l} + x_{i,\epsilon,l+1,k}) \tag{2u}$$
$$y^{ve}_{i,\epsilon,h,k,t} \leq \frac{1}{2}(x_{i,t,h,k} + x_{t,\epsilon,lch(t),rch(t)}) \tag{3u}$$
$$x_{i,j,h,k} \leq \sum_{l=h}^{k-1} y^{ho}_{i,j,h,l,k} + \sum_{t \in an(j)-\{i\}} y^{ve}_{i,j,h,k,t} \quad \text{for all } i,\ j \in I(T_{i,\epsilon,h,k}),\ h \leq k \in ch(i) \tag{2-3t}$$
$$y^{ho}_{i,j,h,l,k} \leq \frac{1}{2}(x_{i,\epsilon,h,l} + x_{i,j,l+1,k}) \tag{2t}$$
$$y^{ho}_{i,j,h,l,k} \leq \frac{1}{2}(x_{i,j,h,l} + x_{i,\epsilon,l+1,k}) \tag{2t'}$$
$$y^{ve}_{i,j,h,k,t} \leq \frac{1}{2}(x_{i,t,h,k} + x_{t,j,lch(t),rch(t)}) \tag{3t}$$
$$z_u \geq \frac{1}{n} \sum_{\{i,j,h,k:es(T_{i,j,h,k})=u\}} x_{i,j,h,k} \quad \text{for all distinct Euler strings } u \text{ of } T_{i,j,h,k} \tag{4}$$

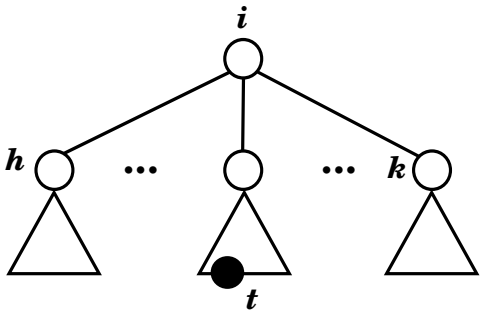


Figure 2.3: Example of ordered tree $T_{i,t,h,k}$.

$$\sum_u z_u \;=\; m \tag{5}$$

where $I(T_{i,\epsilon,h,k})$ denotes a set of internal vertices that are vertices (neither root or leaves) in $T_{i,\epsilon,h,k}$, $an(t)$ denotes a set of ancestors of i ($i \notin an(i)$, and suppose $an(\epsilon) = \emptyset$), and $es(T)$ denotes the Euler string of the ordered tree T (for a tagged tree, the tagged edge with label A is transformed into $Ax\overline{A}$, where x is a special symbol representing the tag). It should be noted that if $es(T) = es(T')$, T is isomorphic to T'.

In the above program, each variable of $x_{i,j,h,k}$, $y^{ho}_{i,j,h,l,k}$, $y^{ve}_{i,j,h,k,t}$, and z_u takes either 0 or 1. Each $x_{i,j,h,k}$ corresponds to subtree $T_{i,j,h,k}$, and $x_{i,j,h,k} = 1$ iff there exists a nonterminal $A_{i,j,h,k}$ in $\mathcal{G}$ that generates $T_{i,j,h,k}$. $z_u = 1$ iff there exists a nonterminal symbol in $\mathcal{G}$ that generates subtree u of $T(V, E)$. The meaning of each (in)equality is as follows (Figure 2.4):

(1u,t) each untagged (tagged) edge $T_{i,\epsilon,j,j}$ ($T_{i,j,j,j}$) must be generated,

(2-3u,t) if $A_{i,j,h,k}$ appears in $\mathcal{G}$ either for at least some l, $A_{i,j,h,k} \to A_{i,j,h,l}A_{i,j,l+1,k} \in \mathcal{G}$, or, for at least some t, $A_{i,j,h,k} \to A_{i,t,h,l}A_{t,j,lch(t),rch(t)} \in \mathcal{G}$ holds. $y^{ho}_{i,j,h,l,k} = 1$ means that $T_{i,j,h,k}$ is horizontally divided at root i into the children $\{s \in ch(i) | s \leq l\}$ and $\{s \in ch(i) | s > l\}$. $y^{ve}_{i,j,h,k,t} = 1$ means that $T_{i,j,h,k}$ is vertically divided at t into two subtrees $T_{i,t,h,k}$ and $T_{t,j,lch(t),rch(t)}$ (if $j \neq \epsilon$, t must be in $an(j)$, otherwise, a divided tree would have two tags),

(4) $A_{i,j,h,k}$ and $A_{i',j',h',k'}$ are identified if both generate the same Euler string u, and

(5) the number of nonterminal symbols used in $\mathcal{G}$ must be m.

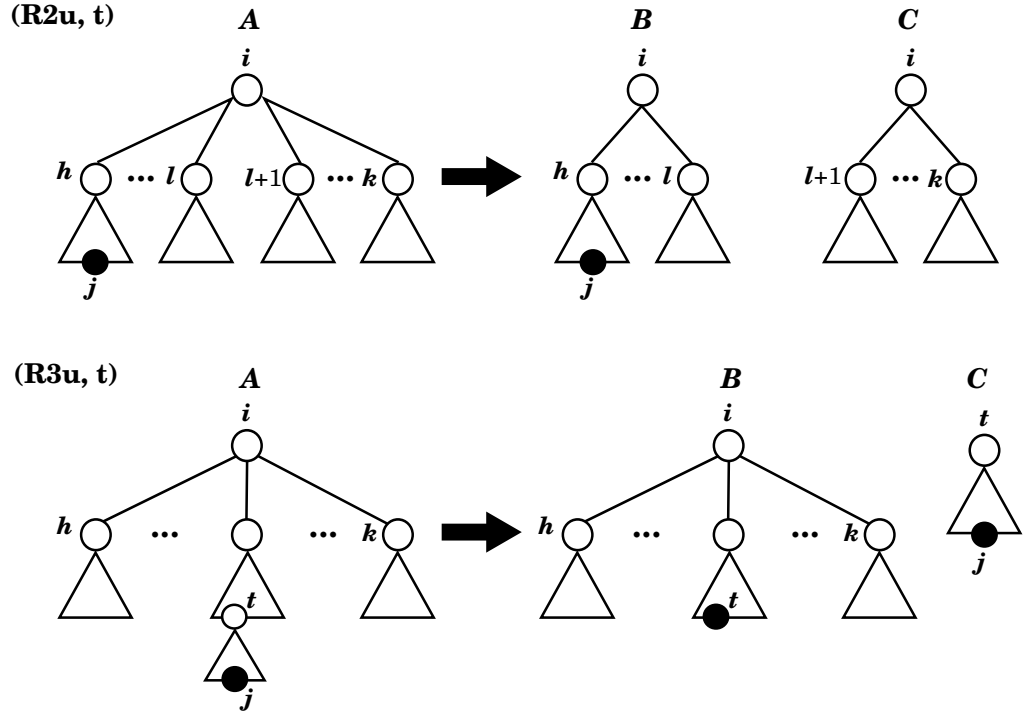


Figure 2.4: Illustration for bisections of the tagged and untagged ordered trees for the production rules.

2.4 Grammar-based unordered tree compression

2.4.1 Minimum EUTG problem

In some cases, a given ordered tree is not well compressed. Figure 2.5 shows an example of such a tree $T(V, E)$, where edges $(1, 2)$,$(1, 3)$,$(1, 6)$,$(3, 4)$, and $(3, 5) \in E$ are labeled with a, c, b, a, and b, respectively. A subtree $T_{3,\epsilon,4,5}$ is the same as a subtree with root i. However, we cannot divide the tree into such a subtree and the remaining part, as in the figure, according to production rules in EOTGs.

Therefore, we need to extend the above IP for the ordered trees to that for the unordered trees. For this purpose, we extend the EOTG to a grammar for the unordered trees, called the elementary unordered tree grammar (EUTG) [2], and use a simple EUTG for rooted unordered tree compression.

A simple EUTG (SEUTG) is defined as 4-tuple $(\Sigma, \Gamma, S, \Delta)$ in a similar way to EOTG. A set of production rules Δ is also the same as that of EOTG (Figure 2.2), except that trees appeared in the production rules are dealt as unordered trees. In other words, there is no sibling relationship between children B and C in the rules

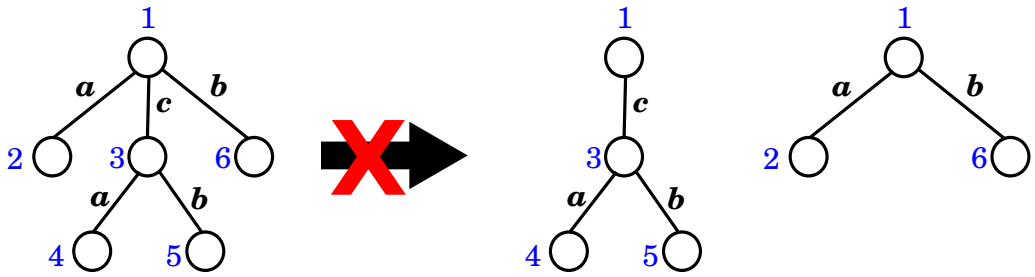


Figure 2.5: Example of an ordered tree that is not well compressed. The left ordered tree cannot be divided into the two right trees. However, this is possible if the left tree is an unordered tree.

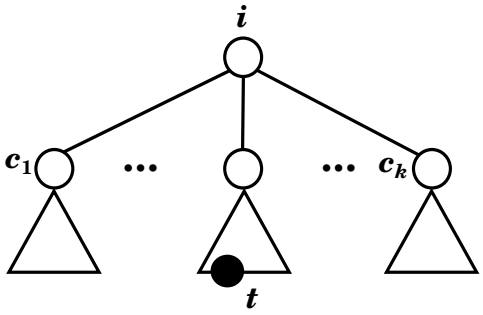


Figure 2.6: Example of the unordered tree $T_{i,t,\mathcal{C}}$. Example of the unordered tree $T_{i,t,\mathcal{C}}$ with a set of children of i, $\mathcal{C} = \{c_1, \cdots, c_k\}$.

(R2u,t). Therefore, we must consider the subtrees of $T(V, E)$ as $T_{i,t,\mathcal{C}}$ (Figure 2.6), where $\mathcal{C}$ ($\neq \emptyset$) is a subset of the children of i. Although $ch(i)$ is considered to be the sequence of children of i for the ordered trees, we allow it be a set of the children of i for the unordered trees.

Thus, within the class of SEUTGs, we define the minimum SEUTG problem as follows:

Minimum SEUTG

Input: Rooted unordered tree $T(V, E)$ and integer m, where V is a set of vertices and E is a set of labeled edges.

Output: Simple EUTG with m nonterminal symbols that generates only T.

2.4.2 IP formulation for minimum EUTG

We must identify unordered subtrees to count the number of nonterminal symbols m. For this purpose, we also use the Euler strings $es(T)$ for the unordered trees T as in the minimum SEOTG. First, the unordered tree T is transformed into the ordered tree T' as follows. The children of each vertex in T are sorted by labels, and if it contains a tag, the tag is moved to the first of the children. Next, $es(T)$ is calculated to be $es(T')$.

Thus, this problem is transformed into the following integer program, where $x_{1,\epsilon,ch(1)} = 1$ holds iff there exists a required EUTG $\mathcal{G}$:

Maximize $x_{1,\epsilon,ch(1)}$
Subject to

$$x_{i,\epsilon,\{j\}} = 1 \quad \text{for all } i,\ j \in ch(i)\ (|ch(j)| = 0) \tag{1u}$$
$$x_{i,j,\{j\}} = 1 \quad \text{for all } i,\ j \in ch(i)\ (|ch(j)| > 0) \tag{1t}$$
$$x_{i,\epsilon,\mathcal{C}} \leq \sum_{\mathcal{C}_1(\neq\emptyset)\subset\mathcal{C}} y^{ho}_{i,\epsilon,\mathcal{C}_1,\mathcal{C}-\mathcal{C}_1} + \sum_{t\in I(T_{i,\epsilon,\mathcal{C}})} y^{ve}_{i,\epsilon,\mathcal{C},t} \quad \text{for all } i,\ \mathcal{C} \subseteq ch(i) \tag{2-3u}$$
$$y^{ho}_{i,\epsilon,\mathcal{C}_1,\mathcal{C}_2} \leq \tfrac{1}{2}(x_{i,\epsilon,\mathcal{C}_1} + x_{i,\epsilon,\mathcal{C}_2}) \tag{2u}$$
$$y^{ve}_{i,\epsilon,\mathcal{C},t} \leq \tfrac{1}{2}(x_{i,t,\mathcal{C}} + x_{t,\epsilon,ch(t)}) \tag{3u}$$
$$x_{i,j,\mathcal{C}} \leq \sum_{\mathcal{C}_1(\neq\emptyset)\subset\mathcal{C}} y^{ho}_{i,j,\mathcal{C}_1,\mathcal{C}-\mathcal{C}_1} + \sum_{t\in an(j)-\{i\}} y^{ve}_{i,j,\mathcal{C},t} \quad \text{for all } i,\ j \in I(T_{i,\epsilon,\mathcal{C}}),\ \mathcal{C} \subseteq ch(i) \tag{2-3t}$$
$$y^{ho}_{i,j,\mathcal{C}_1,\mathcal{C}_2} \leq \tfrac{1}{2}(x_{i,j,\mathcal{C}_1} + x_{i,\epsilon,\mathcal{C}_2})\ (j \in T_{i,\epsilon,\mathcal{C}_1}) \tag{2t}$$
$$y^{ve}_{i,j,\mathcal{C},t} \leq \tfrac{1}{2}(x_{i,t,\mathcal{C}} + x_{t,j,ch(t)}) \tag{3t}$$
$$z_u \geq \tfrac{1}{n}\sum_{\{i,j,\mathcal{C}:es(T_{i,j,\mathcal{C}})=u\}} x_{i,j,\mathcal{C}} \quad \text{for all distinct Euler strings } u \text{ of } T_{i,j,\mathcal{C}} \tag{4}$$
$$\sum_u z_u = m \tag{5}$$

2.5 Results

We implemented the above mentioned IP-based methods for the ordered and unordered trees to perform some computational experiments. All of the experiments were conducted on a PC with a Xeon CPU 3.33 GHz and 10 GB RAM running under the LINUX OS. We used ILOG CPLEX (version 11.2, http://www.ilog.com/products/cplex/) to solve the transformed integer programs.

In the implementation, we first transformed the minimum grammar problem of the ordered and unordered trees into the integer programs. Next, we used ILOG CPLEX, and obtained the number of nonterminal symbols needed in the minimum grammars of this tree compression. Finally, the minimum grammars for the tree compression were constructed from the solution of the IP.

Herein, we selected to use two kinds of datasets: artificial data and the glycan tree-structure data, where the glycans are obtained from KEGG database [35]. As

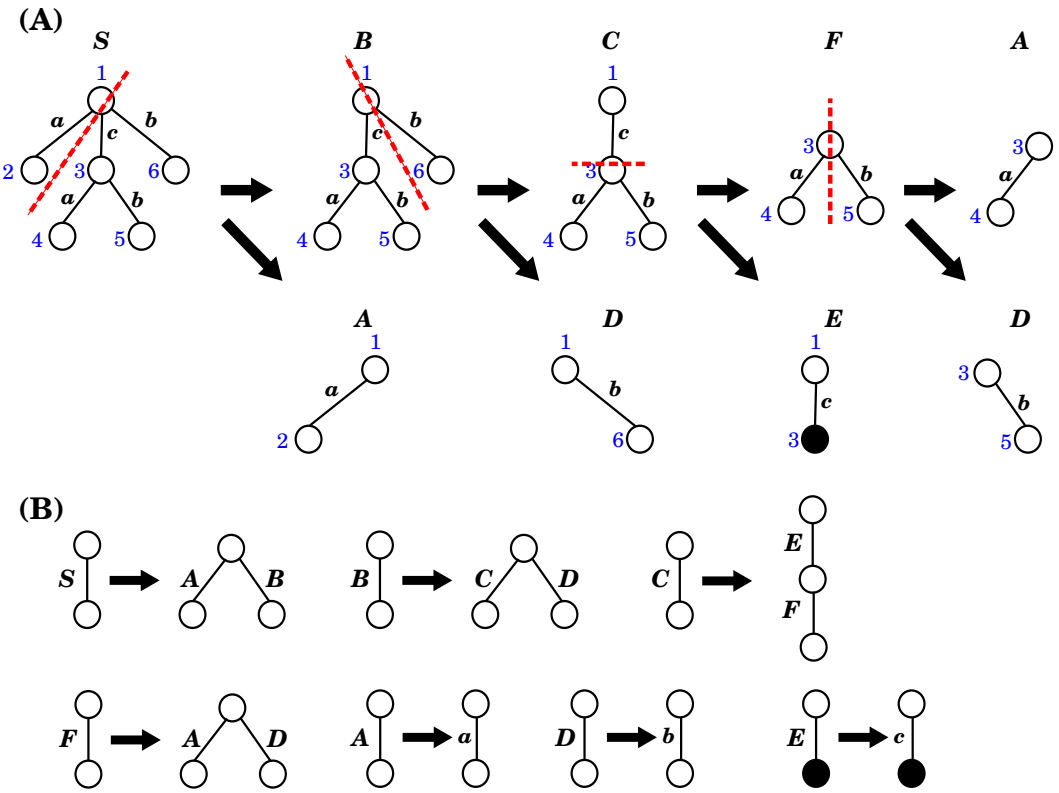


Figure 2.7: Production rules generated by our IP-based method for the minimum SEOTG (A) Derivation of production rules generated by our IP-based method for the minimum SEOTG. (B) Production rules generated by our IP-based method for the minimum SEOTG.

the results, we both tested the computational time for IP solution and compared our performance with that of state-of-the-art method. Further pattern extraction was also carried out to find interesting patterns of glycans.

2.5.1 Instances for tree compression

We chose the left tree T of Figure 2.5 in which edges labeled with "a" and "b" are connected to both endpoints of an edge labeled with "c", and performed computational experiments, where the simple tree T was treated either as an ordered or an unordered tree. When T was regarded as an ordered tree, we generated the integer program with 13 nonterminal symbols for 9 horizontal and 4 vertical divisions. The number of nonterminal symbols needed in the minimum grammar of T is 7 because the number of production rules except (R1u,t) is 4 and the number of terminal sym-

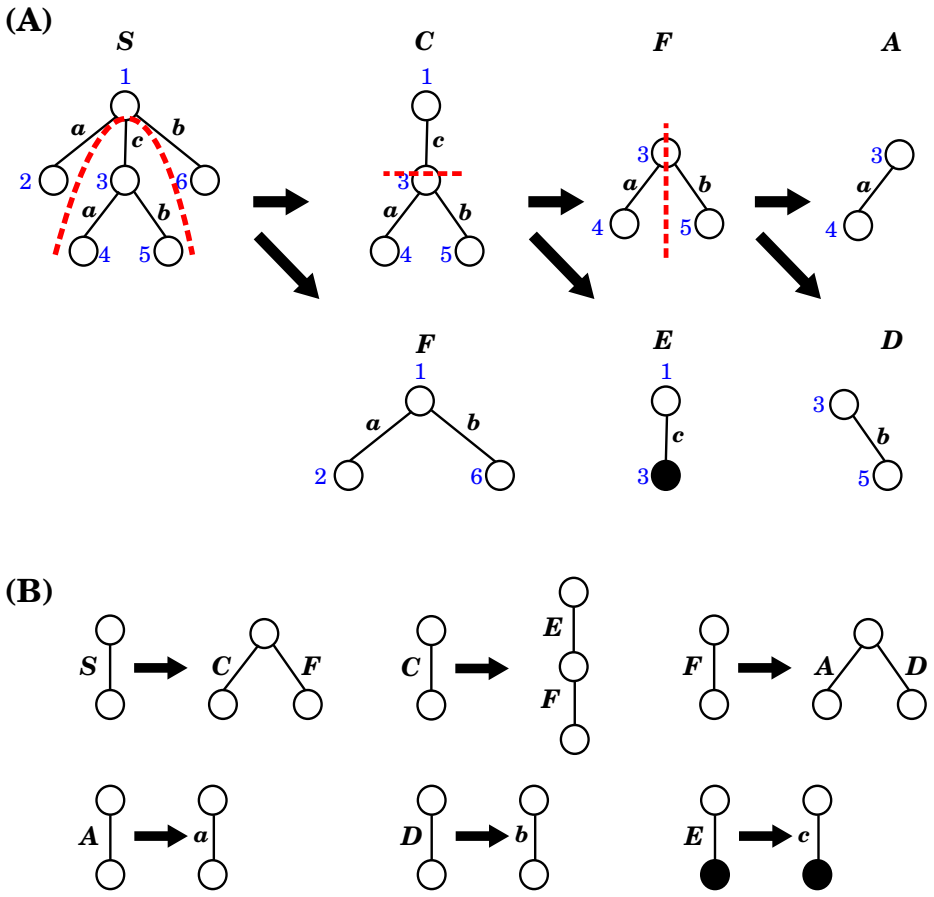


Figure 2.8: Production rules generated by our IP-based method for the minimum SEUTG (A) Derivation of production rules generated by our IP-based method for the minimum SEUTG. (B) Production rules generated by our IP-based method for the minimum SEUTG.

bols is 3. (Figure 2.7, in which nonterminal symbols, S, A, $\cdots$, and F are used). The minimum grammar constructed from the solution of IP is as follows.

$$T_{1,\epsilon,2,6} \rightarrow T_{1,\epsilon,2,2}T_{1,\epsilon,3,6} \quad (2.1)$$
$$T_{1,\epsilon,3,6} \rightarrow T_{1,\epsilon,3,3}T_{1,\epsilon,6,6} \quad (2.2)$$
$$T_{1,\epsilon,3,3} \rightarrow T_{1,3,3,3}T_{3,\epsilon,4,5} \quad (2.3)$$
$$T_{3,\epsilon,4,5} \rightarrow T_{3,\epsilon,4,4}T_{3,\epsilon,5,5}. \quad (2.4)$$

The production rules of this tree compression are also shown in Figure 2.7. The elapsed time to solve the IP was 0.014 s.

When T was regarded as an unordered tree, we generated the integer program

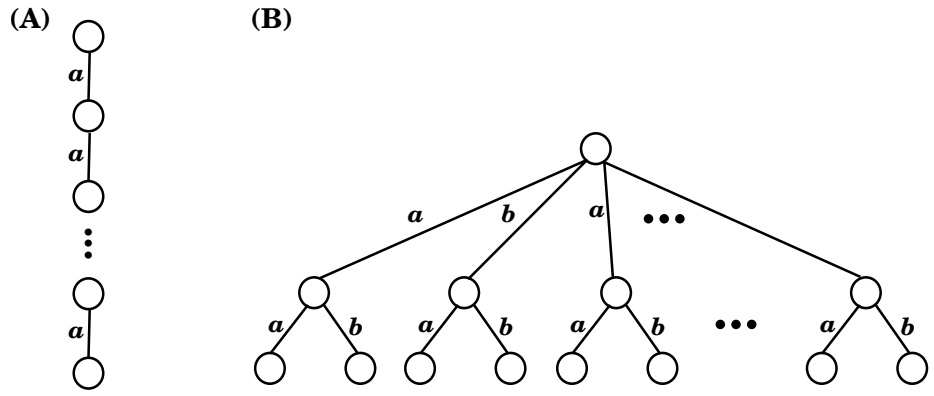


Figure 2.9: Trees used in experiments for evaluation of our IP-based methods (A) Trees having only vertices with degree at the most two. (B) Trees having vertices with degree more than two.

with 14 nonterminal symbols for 12 horizontal and 4 vertical divisions. The minimum number of nonterminal symbols of T is 6 (Figure 2.8). The minimum grammar was constructed as follows.

$$T_{1,\epsilon,2,3,6} \rightarrow T_{1,\epsilon,3}T_{1,\epsilon,2,6} \quad (2.5)$$
$$T_{1,\epsilon,3} \rightarrow T_{1,3,3}T_{3,\epsilon,4,5} \quad (2.6)$$
$$T_{1,\epsilon,2,6} \rightarrow T_{1,\epsilon,2}T_{1,\epsilon,6} \quad (2.7)$$
$$T_{3,\epsilon,4,5} \rightarrow T_{3,\epsilon,4}T_{3,\epsilon,5}. \quad (2.8)$$

The production rules of this tree compression of T are also shown in Figure 2.8. The elapsed time to solve the IP was 0.016 s.

2.5.2 Artificial tree compression

We performed experiments for two types of trees with more vertices (Figure 2.9), where the number of vertices and degree was up to 61 and 20, respectively, and measured the elapsed times.

Type A trees only contain vertices with the degree at most two and edges labeled with a, while Type B trees contain edges labeled with a and b, and the height is two. Table 2.1 shows the results on the elapsed time (seconds) to solve the minimum SEOTG and SEUTG problems by using CPLEX for the ordered and unordered trees of Type A and B with several sizes. m was the same as the minimum number of nonterminal symbols, except the case of Type A trees with 51 vertices. In these cases, CPLEX did not output the solution for $m = 11$ within 8 h. However, we were able to generate the production rules for $m = 12$, although 10 is the minimum

Table 2.1: Results on the elapsed time (seconds) for ordered and unordered trees of type A and B Results on the elapsed time (seconds) for ordered and unordered trees of type A and B (in Figure 2.9) with several sizes. m was the same as the minimum number of nonterminal symbols for each tree, except the case denoted by '*'. '−' denotes that the solver took more than 8 hours.

tree	max	#	ordered		unordered	
type	degree	vertices	m	time	m	time
A	2	11	7	0.021	7	0.019
A	2	31	10	302.74	10	329.20
A	2	41	10	8063.19	10	7730.64
A	2	51	12*	230.51	12*	233.44
B	3	7	9	0.011	8	0.010
B	6	19	11	0.185	10	1.108
B	8	25	11	1.404	10	26440.01
B	10	31	12	2.265	−	−
B	16	49	11	481.15	−	−
B	20	61	13	432.72	−	−

number of nonterminal symbols. If we do not need the minimum grammar, then we can obtain the production rules faster than in the case of finding the minimum grammar.

Furthermore, the results show that the elapsed time for an ordered Type A tree was almost the same as that for the corresponding unordered tree, and the time for an ordered Type B tree was shorter than that for the corresponding unordered tree. Even for the ordered tree with 61 vertices, the time was a few minutes. These results suggest that our proposed method is efficient for ordered trees. We also find that the IP-based method for unordered trees should be used when sibling relationships do not have any meanings and the number of vertices and the maximum degree are not so large because the minimum SEUTG size is always smaller than the minimum SEOTG size. However, if the maximum degree is large and sufficient time is not given, the IP-based method for ordered trees should be used. It is because solving the minimum SEUTG problem for such trees may take too much time whereas the method for ordered trees is expected in many cases to provide a small grammar whose size is close to or the same as that of the smallest grammar obtained by the method for unordered trees.

2.5.3 Glycan tree-structure pattern extraction

It is known that glycans play important roles in a cell such as cellular adhesion and antigen-antibody reaction. Therefore, it is important to analyze structures of glycans. Hizukuri *et al.* extracted characteristic functional motifs of glycans, predicted a leukemia specific glycan motif, and confirmed by biological experiments that the *Agrocybe cylindracea* galectin specifically recognized human leukemic cells [39]. Thus, it is also important to find motifs and repeated patterns of glycans.

Table 2.2: Results of twelve glycans on the grammar size and the elapsed time (seconds) by our proposed IP-based methods for both the minimum SEOTG and SEUTG problems, and TREE-BISECTION [2].

glycan	max	#	# distinct	Min SEOTG		Min SEUTG		TREE-BISECTION	
	degree	vertices	labels	size	time	size	time	size	time
G02703	3	26	3	22	3.68	22	2.9	32	0.001
G03655	3	34	3	47	0.96	47	2.32	49	0.001
G03710	3	28	3	20	0.47	20	0.51	20	0.001
G04045	3	36	3	20	1.77	20	1.98	22	0.001
G04458	3	21	2	16	1.55	16	0.69	36	0.001
G04666	3	20	4	25	1.41	25	0.94	33	0.001
G04859	3	19	5	27	0.12	27	0.25	29	0.001
G05058	3	25	5	26	3.03	26	66.28	36	0.001
G05256	3	25	2	19	3.14	19	3.98	29	0.001
G05552	3	19	5	27	0.66	23	0.23	27	0.001
G06867	3	28	3	22	2.22	22	6.46	26	0.001
G09054	4	31	5	29	2.81	29	6.71	29	0.001

We obtained twelve glycans, G02703, G03655, G03710, G04045, G04458, G04666, G04859, G05058, G05256, G05552, G06867, and G09054 as rooted trees from the KEGG Glycan database [35]. We labeled each edge with a lower-case letter corresponding to the type of sugar of the lower endpoint, because the edges are not labeled in the original data. For each glycan, the maximum degree, the number of vertices, and the number of distinct labels are shown in Table 2.2. Then, we applied our proposed IP-based method for SEUTG to each glycan as an unordered tree, and obtained the production rules. Figures 2.10, 2.11, 2.12, and 2.13 show extracted patterns from the production rules of G03655, G04458, G04666, and G05058. We can see from the result of the generated production rule that the tree of G03655 contains 2 of the same subtrees with 4 vertices and 3 of the same subtrees with 5 vertices, the tree of G04458 contains 2 of the same subtrees with 8 vertices and the subtree contains 3 of the same subtrees with 3 vertices, the tree of G04666 contains 3 of the same subtrees with 5 vertices and 2 of the same subtree with 3 vertices, and the tree of G05058 contains 3 of the same subtrees with 6 vertices and 2 of the same subtrees with 5 vertices. We were able to extract patterns similar to those of

G03655, G04458, G04666, G05058 for the other glycans. The detailed derivation diagrams of production rules for the four glycans are available on our supplementary web site (http://sunflower.kuicr.kyoto-u.ac.jp/morihiro/treegram/).

We compared the results of the grammar size for the minimum SEOTG and SEUTG by our methods with those of an existing method, TREE-BISECTION [2]. TREE-BISECTION repeatedly divides a given tree horizontally and vertically such that the size of a divided subtree is similar to that of another subtree until each subtree consists of an edge. It is known that TREE-BISECTION computes in polynomial time a simple EOTG of size $O(mn^{5/6})$ [2], where m is the size of the minimum simple EOTG and n is the number of vertices of the given tree. Table 2.2 shows the results of the grammar size and the elapsed time by our proposed IP-based methods for the minimum SEOTG and SEUTG problems, and TREE-BISECTION. The minimum SEOTG size was the same as that of the minimum SEUTG for each glycan except G05552 because the tree contains vertices only with at most two children, and all subtrees of a vertex having three children are isomorphic. The size of the grammar generated by our methods was always smaller than or equal to that by TREE-BISECTION, and the ratio was 1.0 (G09054) to 2.25 (G04458). This result shows that our proposed method performs better with the compression ratio than TREE-BISECTION.

2.6 Discussion

We proposed integer programming-based methods for finding the minimum grammars to generate given strings, ordered trees, and unordered trees. By conducting computational experiments, we confirmed that our IP formulations work correctly. The results also show that our IP-based grammar compression is efficient for ordered trees, although some improvements are required for unordered trees.

We applied our proposed method to glycan tree-structure data, and extracted interesting patterns. Although these patterns were obtained from production rules generated for a single tree, we may be able to extract common patterns and rules from multiple glycans by extending our methods to find minimum grammars to generate given forests.

In this framework, we dealt with grammars for trees. However, real structured data often contain some cycles. Therefore, we are in the process of developing IP-based methods for more complex structured data.

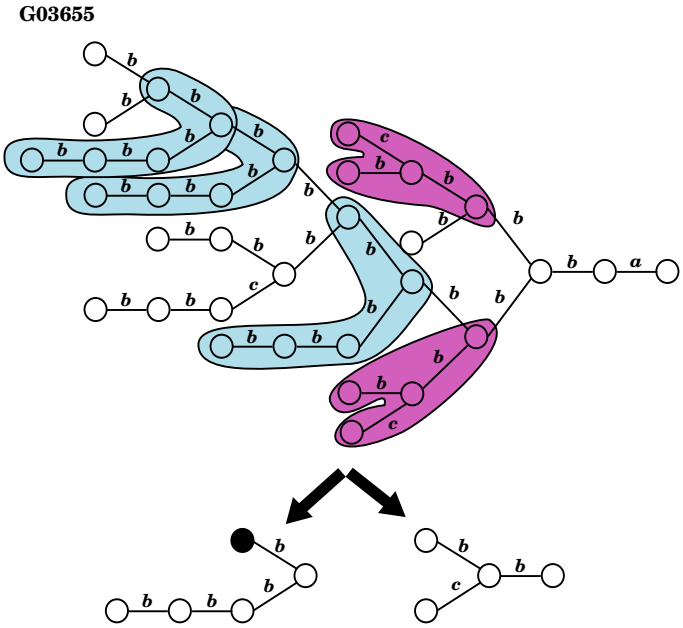


Figure 2.10: Extracted patterns from glycan G03655 The label related with the lower endpoint is attached to each edge. Labels, a, b, and c denote GlcNAc, Man, and P, respectively.

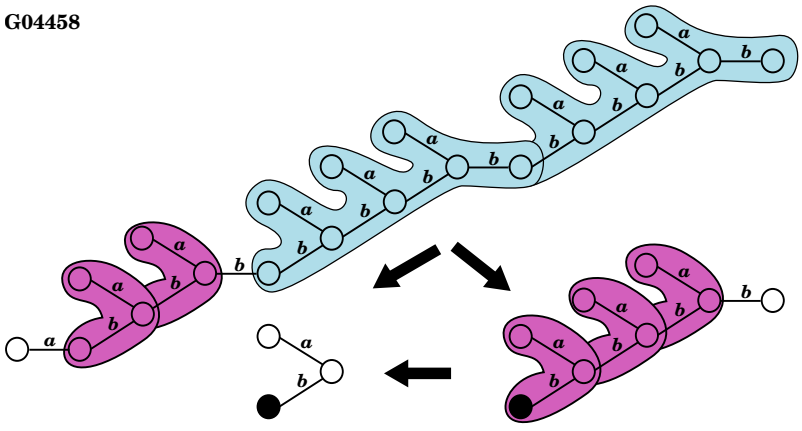


Figure 2.11: Extracted patterns from glycan G04458 The label related with the lower endpoint is attached to each edge. Labels, a, and b denote Xyl, and Glc, respectively.

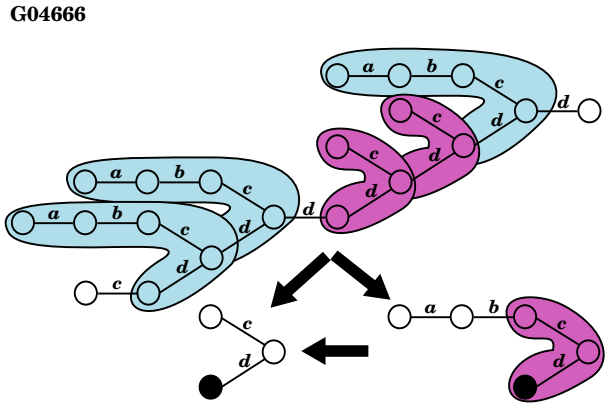


Figure 2.12: Extracted patterns from glycan G04666 The label related with the lower endpoint is attached to each edge. Labels, a, b, c, and d denote LFuc, Gal, Xyl, and Glc, respectively.

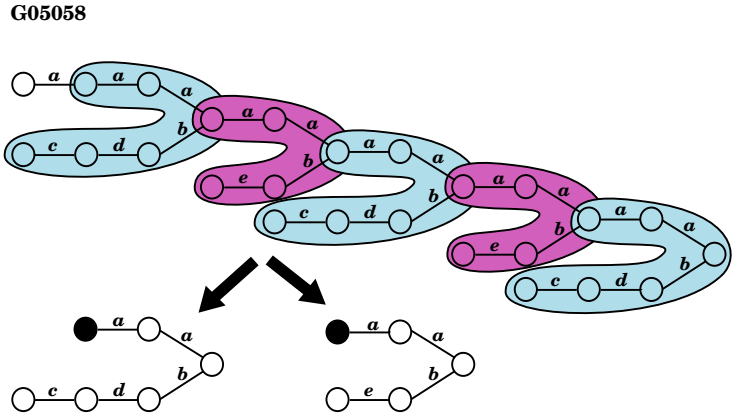


Figure 2.13: Extracted patterns from glycan G05058 The label related with the lower endpoint is attached to each edge. Labels, a, b, c, d, and e denote Glc, Man6Ac, Man, GlcA, and 3-en-eryHexA, respectively.

Chapter 3

Enumeration of tree-like chemical compounds

3.1 Background

Analysis of chemical compounds has gained more attentions recently in bioinformatics because drug design is one of the major goals of bioinformatics and chemical compounds play an important role in metabolic networks. Among various topics on chemical compound analysis, molecular enumeration is a fundamental one that has many applications in the field of drug design [24], and has been studied by mathematicians, computer scientists and chemists for more than one century. Especially, molecular enumeration algorithms have been developed for molecular design, molecular classification and structure elucidation using spectrometric techniques such as mass-spectrum (MS) and nuclear magnetic resonance (NMR) [64]. Although such approaches are not yet as active as marketing to present pharmaceutical service, they still play an essential role in pharmaceutics and therapeutics such that many researchers are involved in developing tools for drug design. It is to be noted that molecular enumeration has also been used as an engine of data mining and knowledge discovery from chemical compound data [19, 40, 44].

Since the first system for enumerating molecules, DENDRAL [69] came out, more researchers in academia have focused on developing computer-aided technology to study this crucial problem. Approaches for enumerating molecules are based on the representation of chemical compounds as molecular graphs. Conceptually, a molecular graph is defined as a connected multi-graph with vertices and edges colored by the atomic symbols and chemical bonds, where the degree of a vertex represents the atomic valence and the multiplicity of an edge represents the bond order [25]. Given molecular formula together with specific restrictions, desired molecules for biological system are enumerated by constructing all distinct graph structures. As

proved by Dobson [20], the solution space for enumerating the desired molecules is estimated to be exponentially increasing as the size of molecules grows. The large search space leads to the foremost barrier for marketing these types of systems to the real-world service.

To date, some algorithms have been developed such as Molgen [25, 31] and Enumol [4, 27, 42, 77], etc. Molgen is known as the popular and useful tool that has been developed since 1985. This integrated project produces a fundamental structure generator that can enumerate desired molecules by given molecular formula with optional further restrictions, e.g. presence or absence of particular substructures. Although functional for constructive structure generation and application-oriented for molecular structure elucidation are continually upgraded by Faulon et al. [25], its enumeration algorithm still requires a vast amount of computational expense.

Recent approaches showed that it is possible to infer trees from feature vectors under constant levels in polynomial time [3, 41]. However, these algorithms are not practical or cannot perform enumeration. One other constructive approach, Enumol was recently proposed [27, 42, 77] that enumerates tree-like molecular graphs by depth-first search (DFS) order. Herein, tree-like compounds have simple structures that can be represented by molecular tree graphs. They defined unique centroid and used the concept of *left-heavy* for labeling of a molecular tree graph such that these utilizations are then shown to be useful for reducing the search space. Although the results showed their competitive advantage to some existing approaches (e.g. Molgen), the growth of computational consumption during the constructive generation step is still needed to be retained.

In this study, we firstly propose efficient algorithms BfsSimEnum and BfsMulEnum for tree-like molecular enumeration by breadth-first search (BFS) order. Unlike algorithms by DFS order [27, 42, 77], our methods enumerate tree structures by keeping their balance which can efficiently save CPU time. For further reduction of the search space, we adjust some important concepts such as *center-rooted*, *left-heavy* and *normal form* when labeling a tree-structured molecular graph. It is interesting to note that target tree graphs are enumerated by BFS order while family tree is searched by DFS order in this study. Finally, computational experiments are performed whose results indicate that our methods are exact and more efficient than state-of-the-art ones. Although we focus on enumeration of tree-like chemical compounds in this study, there are substantial possibilities of extensions of the methods to deal with more general structures. In the conclusion section, we discuss these potential extensions of our algorithms for the future step.

3.2 Preliminaries

Molecular formulas can be represented as molecular graphs whose vertices are labeled with the kinds of the corresponding atoms and edges are labeled with the types of bonds. In this section, we provide some elementary definitions that will be used later in our algorithms.

3.2.1 Molecular trees

We call acyclic connected molecular graphs without multi-edges *simple trees* which represent chemical compounds without multiple bonds. Conversely, *multi-trees* are allowed to have multi-edges.

Let $\Sigma = \{l_1, l_2, \ldots, l_s\}$ be the set of labels of atomic symbols. The degrees of vertices in such a molecular graph are restricted by a valence function $val : \Sigma \to Z^+$ that links each l_i ($l_i \in \Sigma$) with a positive integer. It is noted that only double and triple bonds are taken into account in this study. Herein, we give the label set Σ an order so as to distinguish atoms having the same valence. A *rooted* tree is defined as a tree with one vertex chosen as its root. Then, a molecular tree can be represented as a rooted ordered multi-tree $T(V, E)$, where V is a nonempty finite set of vertices that correspond to atoms, E is a nonempty finite set of edges that correspond to bonds (see an example in Figure 3.1). Let $num(l_i)$ be the number of vertices labeled as l_i in T. Let $height(T)$ be the height of T, and $maxpath(T)$ be a set of the longest paths in T, respectively. Let $T(v)$ denote the subtree rooted at vertex v. Let $l(v)$, $depth(v)$ and $degree(v)$ be the label, depth and degree of vertex v in T, respectively. Let $mul(u, v)$ be the multiplicity of edge (u, v), where u and v are distinct vertices in T.

We define the tree-like compound enumeration problem as follows.

Problem 1. Given a set Σ of s labels representing atoms, number n_i of each label l_i, a valence function $val : \Sigma \to Z^+$, enumerate all molecular multi-trees T such that $n_i = num(l_i)$ for all l_i in Σ and $degree(v) = val(l(v))$ for all vertices $v \in T$.

3.2.2 Center-rooted

We define a unique *center* to a rooted tree T as the center of any path in $maxpath(T)$, where such a center should be a single vertex (Figure 3.2(a)) or an edge (Figure 3.2(b)). It is obvious that such a center in T is unique regardless of the number of elements in $maxpath(T)$. Thus T is called *center-rooted* if its root is the center or one endpoint of the center.

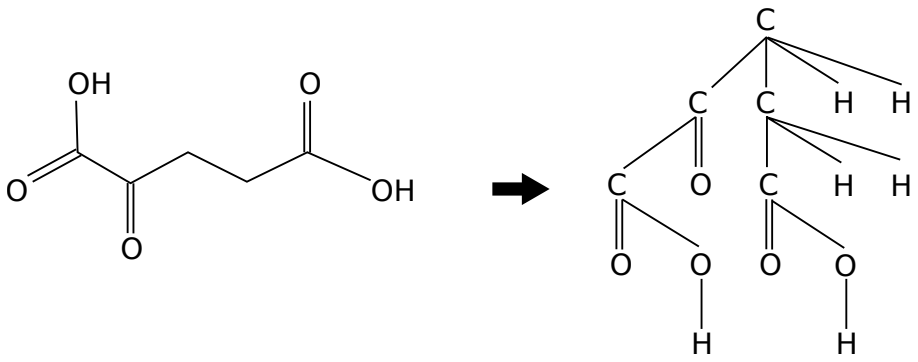
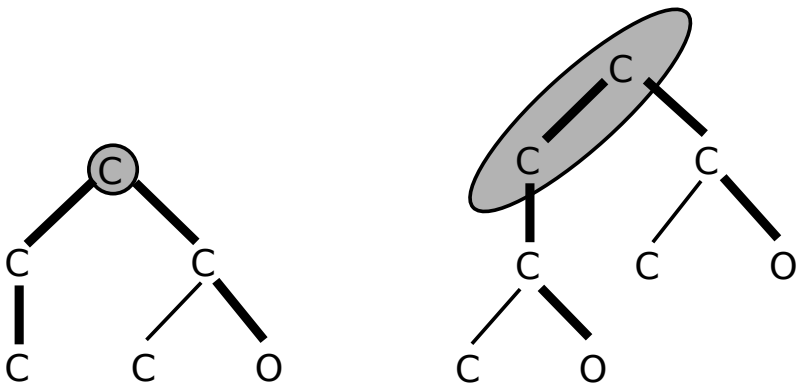


Figure 3.1: Example of transforming 2-oxo-glutarate into a rooted ordered multi-tree T. This transformed molecular tree has $depth(T) = 4$, $maxpath = \{\text{HOCCCCCOH}\}$ and label set {C, O, H}, where $num(\text{C}) = 5$, $num(\text{O}) = 5$ and $num(\text{H}) = 6$.



(a) (b)

Figure 3.2: Illustration of two kinds of center-rooted trees. The thick lines represent one of the longest paths, and the vertices in circles represent the center.

3.2.3 Left-heavy

We introduce two inequalities $>_s$ and $>_m$ for ordered simple and multi-trees, which are recursively defined as in Definition 1 and Definition 2, respectively. We say that $T(u)$ is heavier than $T(v)$ if $T(u) >_s T(v)$ or $T(u) >_m T(v)$.

Definition 3.2.1. *Let $(u_1, u_2, \ldots, u_h)$ and $(v_1, v_2, \ldots, v_k)$ be the children of u and v in simple or multi-tree T, respectively. We define $T(u) >_s T(v)$ if*

1. *$l(u) > l(v)$ (see Figure 3.3(a)), or*
2. *$l(u) = l(v)$ $\exists\, i$, $\forall j \leq i$ $T(u_j) =_s T(v_j)$, and*
   - *(a) $i < \min\{h, k\}$, $T(u_{i+1}) >_s T(v_{i+1})$, or*
   - *(b) $i = k < h$ (Figure 3.3(b)).*

   Specially, we define $T(u) =_s T(v)$ if $l(u) = l(v)$ and $T(u_j) =_s T(v_j)$ for all $j \leq h = k$.

Definition 3.2.2. *We define $T(u) >_m T(v)$ for multi-tree T if*

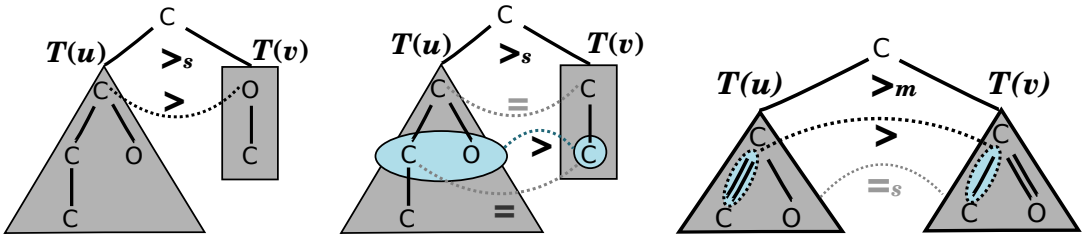
1. *$T(u) >_s T(v)$, or*
2. *$T(u) =_s T(v)$ (see also Figure 3.3(c)), and*
   $\exists i$, $(\forall j \leq i)$ $mul(e_j) = mul(e'_j)$ and $mul(e_{i+1}) > mul(e'_{i+1})$, where $e_1, e_2, \ldots, e_m$ (resp., $e'_1, e'_2, \ldots, e'_m$) be all the edges in $T(u)$ (resp., $T(v)$) in the BFS order. Specially, we define $T(u) =_m T(v)$ if $T(u) =_s T(v)$ and $mul(e_j) = mul(e_j)$ for all $j \leq m$.

For reducing the search space in the generation process, we utilize the definition of *left-heavy* for rooted trees, where the definition is slightly modified from that by Fujiwara et al. [27] and Nakano and Uno [61].

Definition 3.2.3. *A molecular tree T is left-heavy if $T(v_i) \geq_m T(v_{i+1})$ $(i = 1, \ldots, k-1)$ holds for the children $(v_1, \ldots, v_k)$ of each vertex v in T.*

3.2.4 Normal form

In order to avoid duplications, we utilize a notion of *normal form* to molecular trees as formalized in Definition 4. The normal form includes the ideas of center-rooted and left-heavy. We call a tree in the normal form a *normal tree*.



(a) (b) (c)

Figure 3.3: Illustration of inequality $T(u) >_s T(v)$ and $T(u) >_m T(v)$. Substructures in gray areas represent comparative subtrees rooted at u and v in each subfigure. The label set of these examples is $\Sigma = \{C, O\}$ whose order is $C > O$. (a) $T(u) >_s T(v)$ holds since the root of $T(u)$ has a greater label than that of $T(v)$. (b) Comparison of their corresponding descendants in BFS order shows that $T(u) >_s T(v)$. (c) The special case of $T(u) >_m T(v)$ with $T(u) =_s T(v)$, for which further comparison is needed to check their corresponding edge multiplicity in BFS order. Since the edge multiplicity in dot circles of $T(u)$ is greater than the corresponding location of $T(v)$, $T(u) >_m T(v)$ holds.

Definition 3.2.4. *Let T be a left-heavy center-rooted ordered tree rooted at r.*

1. *If the center is a single vertex, then T is a normal tree.*
2. *If the center is an edge (r, v) and $T_r(v) \geq_m T_v(r)$, where $T_r(v)$ ($T_v(r)$) denotes the subtree induced by v (r) and its descendants when r (v) is root, then T with root r is a normal tree (see also an example in Figure 3.4).*

3.2.5 Family tree

Our approach searches a special tree structure called *family tree*. Let $\mathcal{C}_n$ denote a set of left-heavy center-rooted simple trees with at most n vertices, where $n = \sum_{i=1}^{s} n_i$. Suppose that a tree $T \in \mathcal{C}_n$ has k vertices ($0 < k \leq n$), numbered as $(v_1, v_2, \ldots, v_k)$ in BFS order. Let $P(T)$ be a tree generated from T by removing the last vertex v_k of T. We call $P(T)$ the *parent* of T.

Theorem 3.2.5. *If a given tree T is left-heavy center-rooted, then its parent $P(T)$ is left-heavy center-rooted as well.*

Proof. Suppose that v_k is the last vertex of T in BFS order. In terms of Definition 3, for any $T(v_j)$ in T such that $v_k \in T(v_j)$, we have $T(v_h) \geq_m T(v_j) >_m T(v_l)$, where v_h denotes any left sibling of v_j and v_l denotes any right sibling of v_j, respectively.

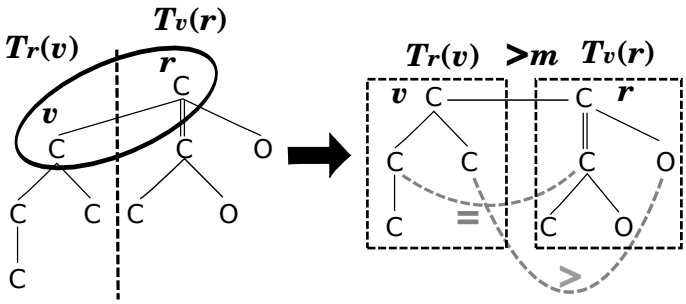


Figure 3.4: Illustration of determining a normal form. Since the definition of normal form is based on the ideas of left heavy and center-rooted, the root is one endpoint of the center, further comparison is needed between subtree $T_r(v)$ and $T_v(r)$, only if r is the root and v is the other endpoint of the center. The given tree is determined as a normal tree because $T_r(v) >_m T_v(r)$.

It is noted that $T(v_j) \neq T(v_l)$ always holds, because $height(T(v_j)) > height(T(v_l))$ to keep v_k the last vertex. Then, $T(v_h) >_m T(v_j) - v_k \geq_m T(v_l)$ holds for such $T(v_j)$ (see also an illustration in Figure 3.5). Together with the definition that $P(T)$ is a tree with one less of the last vertex than T, $P(T)$ is also left-heavy.

The removed vertex v_k is in a path of $maxpath(T)$ because v_k is one of the deepest vertices in T. Let u be another endpoint of the path. Then, $depth(u) = depth(v_k)$ or $depth(u) = depth(v_k) - 1$ holds because T is center-rooted. Correspondingly, we obtain that $depth(u) = depth(parent(v_k)) + 1$ or $depth(u) = depth(parent(v_k))$. If the path between u and $parent(v_k)$ belongs to $maxpath(P(T))$, then $P(T)$ is center-rooted because the difference between $depth(u)$ and $depth(parent(v_k))$ is at most 1 (Figures 3.6(b) and 3.6(d)). Otherwise, $P(T)$ is still center-rooted because there exist other paths than the path between u and v_k in $maxpath(T)$ and these paths also belong to $maxpath(P(T))$ (Figures 3.6(a) and 3.6(c)). Thus, $P(T)$ is center-rooted as well. □

Theorem 1 implies that for any $T \in \mathcal{C}_n$, its parent tree $P(T)$ belongs to $\mathcal{C}_n$. Similarly, we can generate $P(P(T))$ by removing the vertex v_{k-1}, the deepest rightmost leaf of $P(T)$, from $P(T)$. Thus, a unique sequence T, $P(T)$, $P(P(T))$, $P(P(P(T)))$, $\ldots$, ϕ of trees in $\mathcal{C}_n$ can be generated by repeatedly removing the deepest rightmost leaf for each T in $\mathcal{C}_n$. A *family tree* of $\mathcal{C}_n$, denoted by $\mathcal{F}_n$, is defined by merging all these sequences. Obviously, each vertex in such $\mathcal{F}_n$ represents a tree in $\mathcal{C}_n$.

Furthermore, a family tree for molecular multi-trees can be similarly defined. Let $\mathcal{S}_m$ denote a set of left-heavy center-rooted multi-trees with at most m multi-edges. Suppose that a tree T belongs to $\mathcal{S}_m$ with h multi-edges ($0 < h \leq m$), and $(e_1, e_2, \ldots, e_h)$ is a sequence of multi-edges of T in BFS order. Let $P(T)$ be a tree

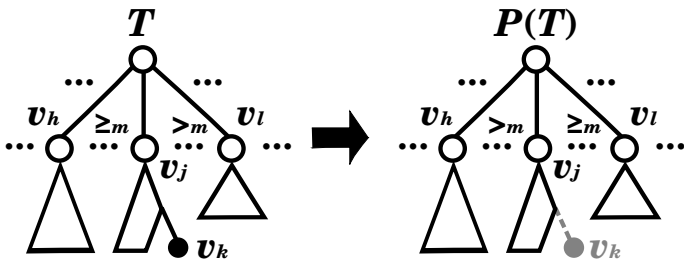


Figure 3.5: Illustration of $P(T)$ being left-heavy. Suppose that v_k is the last vertex of T in BFS order, $T(v_j)$ is any tree such that $v_k \in T(v_j)$, and v_h and v_l are any left and right siblings of v_j in T. Since T is left-heavy and $T(v_h) \geq T(v_j) > T(v_l)$ holds, $T(v_h) \geq T(v_j) - v_k \geq T(v_l)$ always holds (the right-side).

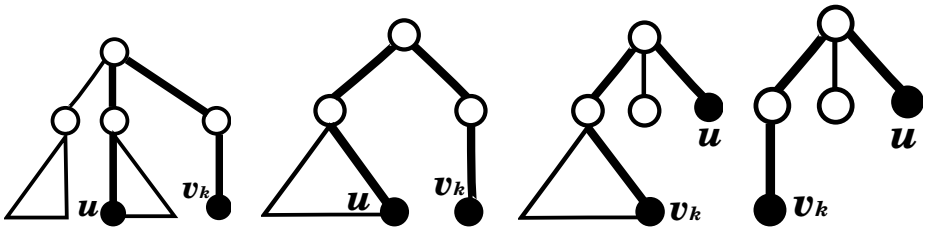
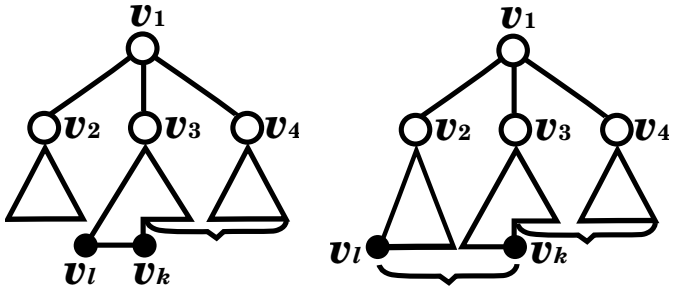


Figure 3.6: Illustration of 4 kinds of center-rooted trees with their longest paths including the deepest rightmost vertex v_k (i.e., last vertex in BFS order). The black vertices denote the deepest rightmost vertices and the other endpoints of the longest paths in these trees.

generated from T by changing the multi-edge e_h to be a single edge. Then $P(T)$ is called the *parent* of T with one less multi-edges of T. We can easily prove that Theorem 1 can be extended to multi-trees and $P(T)$ for each T in $\mathcal{S}_m$ is in $\mathcal{S}_m$ as well. Similarly, $P(P(T))$ can be generated by changing the multi-edge e_{h-1} to a single edge from $P(T)$. Thus, a unique sequence T, $P(T)$, $P(P(T))$, $P(P(P(T)))$, ..., of trees in $\mathcal{S}_m$ can be generated by repeatedly changing the last multi-edge in BFS order to a single edge for each T in $\mathcal{S}_m$. A family tree of $\mathcal{S}_m$, denoted by $\mathcal{F}_m^M$, is defined by merging all these sequences. Obviously, each vertex in $\mathcal{F}_m^M$ represents a tree in $\mathcal{S}_m$ and root of such $\mathcal{F}_m^M$ is a simple tree. It should be noted that both $\mathcal{F}_m$ and $\mathcal{F}_m^M$ are searched by DFS order.



(a) (b)

Figure 3.7: Illustration for determining the possible positions where a new leaf should be added to a current tree. (a) If the deepest leftmost vertex v_l and the deepest rightmost vertex v_k are in the same subtree, a new leaf can be added to only vertices from $parent(v_k)$ to v_{l-1} (within the brace). (b) If v_l and v_k are included in distinct subtrees, a new leaf can be added to vertices from $parent(v_k)$ to v_k (within the brace).

3.3 Methods

In this section, we propose BfsSimEnum and BfsMulEnum for enumerating molecular simple and multi-trees by breadth-first search order. As Ishida et al. [42] and Shimizu et al. [77] pointed out, the number of enumerated solutions exponentially increases with the increasing number of atoms. To reduce the large search space, concepts of center-rooted, left-heavy and normal form are taken in use as restrictions for avoiding duplicates.

3.3.1 BfsSimEnum for simple tree enumeration

Given a molecular formula with valence function, while trying to enumerate all possible simple trees, BfsSimEnum searches a family tree that: each vertex is a left-heavy and center-rooted simple tree, and specially, each leaf is in normal form. Notice that each vertex in such a family tree represents a tree with at most n' vertices, where n' is the total number of atoms whose valences are greater than 1, since atoms with valence one such as hydrogen atoms can be easily added as leaves at last.

This algorithm tries to search a family tree which starts with an empty tree, and grows up by repeatedly adding a new vertex to a current tree T in BFS order at every turn until all n' vertices are added. Algorithm 1 gives an introduction of BfsSimEnum (see also an illustration in Figure 3.9).

Firstly, BfsSimEnum constructs a tree T with one single vertex whose valence

is greater than 1. As an addition step, this algorithm repeatedly adds a new vertex to possible positions of T according to left-heavy and center-rooted to construct a new tree. BfsSimEnum outputs a generated tree if and only if n' vertices are added and it is in normal form. The correctness of BfsSimEnum can be seen as follows. This algorithm searches all possible left-heavy center-rooted simple trees of a family tree which can cover all solutions of Problem 1. Then duplicates are excluded by checking the normal forms. Therefore, this algorithm correctly outputs all structures without any repetition.

Input An ordered set of labels $\Sigma = \{l_1, \ldots, l_s\}$, where $l_1 > l_2 \ldots > l_s$, number n_j of each label l_j, a valence function $val : \Sigma \to Z^+$
Output A set of all possible simple trees $\mathcal{R}$ which are *normal trees*
BfsSimEnum(Σ, val, $\{n_i\}$)
  $\mathcal{R} := \emptyset$
  for each $l_j \in \Sigma$ such that $val(l_j) > 1$ **do**
    $T :=$ a tree consisted of a root with l_j
    AddNode(Σ, val, $\{n_j\}$, T, $\mathcal{R}$)
  return $\mathcal{R}$
end
AddNode(Σ, val, $\{n_j\}$, T, $\mathcal{R}$)
  $n' :=$ the number of given atoms whose valences are greater than 1
  if $|T| = n'$ **and** T holds *normal form* **then**
    $\mathcal{R} := \mathcal{R} \cup \{T\}$
  else
    $v_k, v_l :=$ the deepest rightmost and leftmost vertices in T, respectively
    if v_k and v_l are included in the same proper subtree **then**
      $v_e := v_{l-1}$
    else $v_e := v_k$
    for each v_i from $parent(v_k)$ to v_e in BFS order **do**
      if $degree(v_i) < val(l(v_i))$ **then**
        $l_g :=$ the largest possible label for v_{k+1} (see Figure 3.8)
        for each $l_j \in \Sigma$ such that $l_j \leq l_g$ **and** $val(l_j) > 1$ **and** $num(l_j) < n_j$ **do**
          $T' := T$; add a new vertex with l_j as the last child of v_i in T'
          AddNode(Σ, val, $\{n_j\}$, T', $\mathcal{R}$)
end

Algorithm 1: BfsSimEnum for simple tree enumeration.

The point of this algorithm is how to keep a new constructed tree left-heavy and center-rooted at an addition step. Let v_k and v_l be the deepest rightmost and leftmost vertices in T, respectively. To keep a new tree center-rooted, the candidate positions to be added are determined by checking whether or not vertices v_l and v_k are in a subtree: (i) if they are in a subtree, a vertex can be added to only the positions ranging from $parent(v_k)$ to v_{l-1} (see an illustration in Figure 3.7(a)); (ii) otherwise, a vertex can be added to the positions ranging from $parent(v_k)$ to v_k (see also the Figure 3.7(b)). To keep a new tree left-heavy, candidate labels for a new added vertex v_{k+1} are determined by checking subtrees including v_{k+1}. Since the label set Σ is ordered as $l_1 > l_2 > \ldots > l_s$, our algorithm aims to

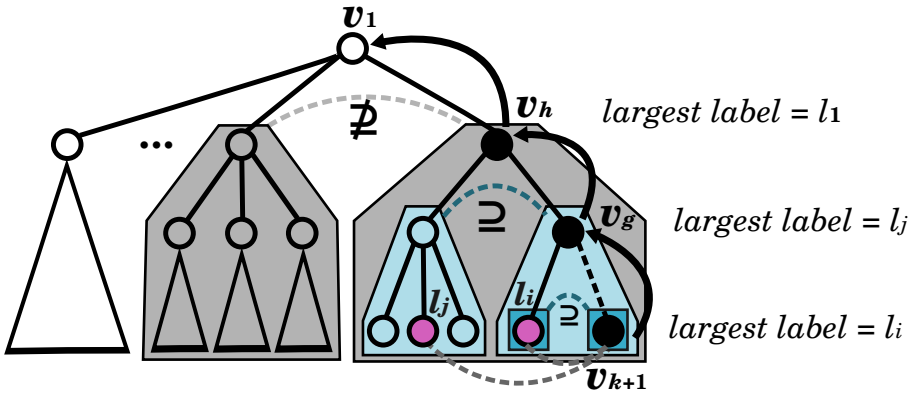


Figure 3.8: Illustration for determining the largest possible label for v_{k+1}. The black circles are vertices having left siblings in the path from v_{k+1} to the root v_1. BfsSimEnum separately compares all of the subtrees rooted at such vertices with the subtrees rooted at their left siblings to determine the largest possible label. For instance, since $T(v_{k+1}) \subseteq T(v_k)$ and $T(v_g) \subseteq T(v_{g-1})$, the largest possible label at these comparison steps are l_i and l_j, which are the labels of corresponding vertices (light gray circles) of v_{k+1} in $T(v_k)$ and $T(v_{g-1})$, respectively; while since $T(v_h) \not\subseteq T(v_{h-1})$, no comparison is done for v_h and the largest possible label is l_1. The largest possible label for v_{k+1} is then determined by using l_i, l_j and l_1.

seek the largest possible label for v_{k+1} such that all smaller ones are candidate labels for v_{k+1}. Suppose that v_h is a vertex in the path from v_{k+1} to the root v_1, which has left sibling. Let $T(v_h) \subseteq T(v_{h-1})$ if and only if there exists an injection mapping ψ of vertices from $T(v_h)$ to $T(v_{h-1})$ such that $l(v_i) = l(\psi(v_i))$ for all $v_i \in V(T(v_h))$ and $(\psi(v_i), \psi(v_j)) \in E(T(v_{h-1}))$ for all $(v_i, v_j) \in E(T(v_h))$, where $V(T)$ and $E(T)$ denote the vertex set and edge set of tree T, respectively. The largest possible label is determined by comparing the subtrees rooted at all such v_h with the subtrees rooted at their corresponding left siblings: (i) if $T(v_h) \subseteq T(v_{h-1})$, the largest label is l_j, where l_j is the label of corresponding vertex of v_{k+1} in $T(v_{h-1})$; (ii) if $T(v_h) \not\subseteq T(v_{h-1})$, the largest possible label is l_1. The largest possible label for v_{k+1} is thus determined as the smallest one obtained from these comparison steps (see an illustration in Figure 3.8).

3.3.2 BfsMulEnum for multi-tree enumeration

We propose BfsMulEnum for multi-tree enumeration, which starts with an output of BfsSimEnum and repeatedly changes a single edge to a multi-edge in BFS order at every turn until all possible multi-edges are added. As mentioned before, only edges

with multiplicity 2 or 3 are taken into account as multiple bonds in this approach.

Let $\mathcal{M}_2$ and $\mathcal{M}_3$ denote the number of double bonds and triple bonds, respectively. $\mathcal{M}_2$ and $\mathcal{M}_3$ are computed so that they satisfy the following equation:

$$2\mathcal{M}_2 + 4\mathcal{M}_3 = \sum_{i=1}^{h} n_i \cdot val(l_i) - 2\sum_{i=1}^{h} n_i - \sum_{j=h+1}^{s} n_j + 2, \tag{3.1}$$

where two sets of labels $\{l_1, \ldots, l_h\}$ and $\{l_{h+1}, \ldots, l_s\}$ represent atoms whose valences are greater than 1 and whose valences are 1, respectively. In the example in Figure 3.9, as the molecular formula is given as $C_2O_2H_2$, feasible multiple bonds for $\mathcal{M}_2$ and $\mathcal{M}_3$ should satisfy that $2\mathcal{M}_2 + 4\mathcal{M}_3 = 2 \cdot 4 + 2 \cdot 2 - 2 \cdot (2+2) - 2 + 2 = 4$, which means that target molecular trees of this example should have two double bonds or one triple bond such that $(\mathcal{M}_2, \mathcal{M}_3) = (2, 0), (0, 1)$.

Input $\mathcal{M}_2, \mathcal{M}_3$ and output $\mathcal{R}$ of Algorithm 1
Output a set of all possible multi-trees $\mathcal{R}_m$
BfsMulEnum ($\mathcal{M}_2, \mathcal{M}_3, \mathcal{R}$)
  $\mathcal{R}_m := \emptyset$
  while $|\mathcal{R}| \neq 0$ **do**
    $T :=$ a tree of $\mathcal{R}$
    remove T from $\mathcal{R}$
    AddMultiedge (T, root of T, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{R}_m$)
  return $\mathcal{R}_m$
end
AddMultiedge (T, v_i, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{R}_m$)
    if $\mathcal{M}_2 = 0$ **and** $\mathcal{M}_3 = 0$ **and** T is a *normal tree* **then**
      $\mathcal{R}_m := \mathcal{R}_m \cup \{T\}$
    else
      AddMultiedge(T, v_{i+1}, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{R}_m$)
      $miss(v_i) := val(l(v_i)) - degree(v_i)$
      $miss(parent(v_i)) := val(l(parent(v_i))) - degree(parent(v_i))$
      if $miss(v_i) \geq 1$ **and** $miss(parent(v_i)) \geq 1$ **and** $\mathcal{M}_2 > 0$ **then**
        $T_2 := T$
        set double bond to $(parent(v_i), v_i)$ in T_2
        AddMultiedge(T_2, v_{i+1}, $\mathcal{M}_2 - 1$, $\mathcal{M}_3$, $\mathcal{R}_m$)
      if $miss(v_i) \geq 2$ **and** $miss(parent(v_i)) \geq 2$ **and** $\mathcal{M}_3 > 0$ **then**
        $T_3 := T$
        set triple bond to $(parent(v_i), v_i)$ in T_3
        AddMultiedge(T_3, v_{i+1}, $\mathcal{M}_2$, $\mathcal{M}_3 - 1$, $\mathcal{R}_m$)
end

Algorithm 2: BfsMulEnum for multi-tree enumeration.

From the output of BfsSimEnum together with $\mathcal{M}_2$ and $\mathcal{M}_3$ determined as above, BfsMulEnum aims to construct a set of target multi-trees $\mathcal{R}_M$ by BFS order, see also the details in Algorithm 2. BfsMulEnum recursively sets a multi-edge to feasible positions of T according to normal form to construct a new tree. Only if all feasible multiple bonds are set, BfsMulEnum outputs such a new tree. Different

from BfsSimEnum, at a setting step, BfsMulEnum needs to check whether a new tree is in normal form by comparing the edge multiplicity together with checking the feasibility of setting other multi-edges when generation for such a new tree is still continued. The correctness of BfsMulEnum can be similarly validated as follows. This algorithm generates all possible multi-trees by keeping them left-heavy which can cover all of the solutions. Finally, all possible structures are correctly enumerated by checking the normal forms.

Although it consumes a little more computational expense for enumerating multi-tree structures than that for enumerating simple ones in this study, computation of BfsMulEnum is not complicated since it only deals with edges without any structural changes. Figure 3.9 illustrates the process of both BfsSimEnum and BfsMulEnum. It should be noted that atoms with valence 1 are added as leaves at last.

3.3.3 Time complexity analysis

From Algorithm 1, we can see that the size of $\mathcal{R}$ (which is used to store enumerated isomers) exponentially grows with the number of atoms increasing. Therefore both space and time complexities of BfsSimEnum are exponential. Since BfsMulEnum uses the outputs of BfsSimEnum, its space and time complexities are exponential as well. Whereas, the space complexity of our methods can be polynomial if we do not store the outputs.

Several researchers have focused on the output polynomial (which means to output one solution in polynomial time). Although the time complexity of our algorithms is exponential, development of output polynomial time algorithms will be our future work.

3.4 Results

Computational experiments were performed on BfsSimEnum and BfsMulEnum using a PC with Xeon CPU 3.47GHz and 24GB memory.

3.4.1 Comparison with existing methods

We assessed the computational performance by comparison with two state-of-the-art methods, Molgen (Version 3.5) and Enumol, under the same computational environment. The results are shown in Table 3.1 and Table 3.2. Unlike existing approaches whose molecular structures are generated by DFS order, the results successfully show that generating a solution by BFS order also performs well or even

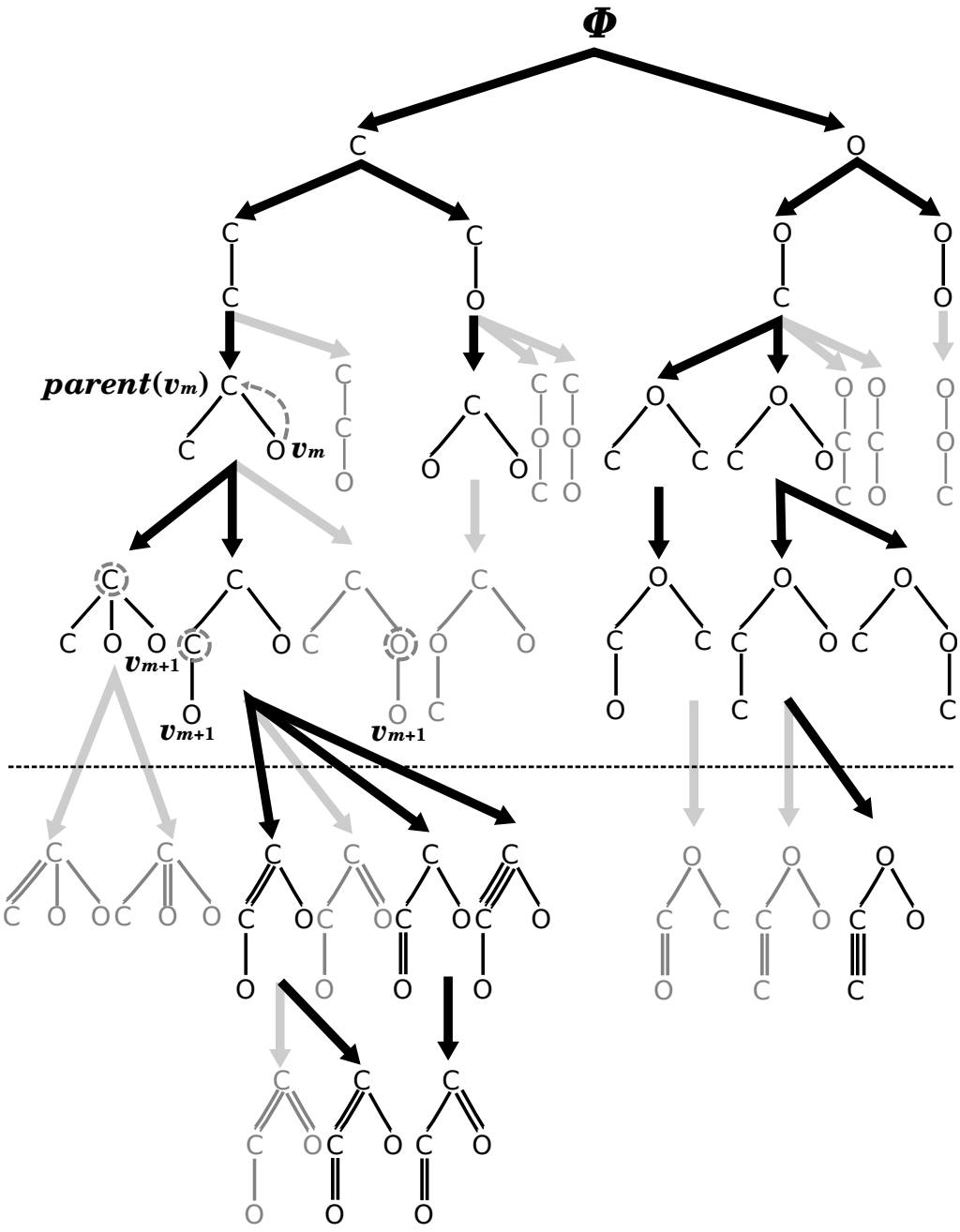


Figure 3.9: Illustration of BfsSimEnum and BfsMulEnum. Algorithm 1 is processed above the dot line; Algorithm 2 is processed below the dot line. Graphs in gray color are considered as invalid by the algorithms and thus are not stored or proceeded any more. It should be noted that Hydrogen atoms are added as leaves at last.

Table 3.1: Comparison of BfsSimEnum with existing methods.

Molecular formula	# Enumerated results	Computational time (Sec.)		
		BfsSimEnum	Molgen	Enumol
$C_{18}H_{38}$	60523	0.016	3.04	0.025
$C_{19}H_{40}$	148284	0.036	5.93	0.060
$C_{20}H_{42}$	366319	0.086	8.18	0.15
$C_{22}H_{46}$	2278658	0.53	79.80	0.939
$C_{24}H_{50}$	14490245	3.284	733.12	6.153
$C_{26}H_{54}$	93839412	21.361	7367.48	41.292
$C_6O_3H_{14}$	772	0.001	0.01	0.001
$C_7O_3H_{16}$	2275	0.002	0.01	0.002
$C_{10}O_4H_{22}$	317677	0.072	1.19	0.108
$C_{12}O_4H_{26}$	3118708	0.691	15.31	1.088
$C_{16}O_4H_{34}$	278960984	60.16	2272.55	101.69
$C_{18}O_4H_{38}$	2567668160	533.84	-	965.4
$C_6N_2O_3H_{16}$	140014	0.031	0.36	0.049
$C_7N_2O_2H_{18}$	82836	0.019	0.17	0.029
$C_7N_3O_2H_{19}$	649970	0.135	1.48	0.216
$C_8N_3O_2H_{21}$	2361374	0.485	6.24	0.81
$C_9N_2O_2H_{22}$	893769	0.188	2.59	0.309
$C_9N_3O_2H_{23}$	8373347	1.683	25.52	2.839
$C_{10}N_3O_2H_{25}$	29105924	5.887	93.94	10.303
$C_{11}N_3O_2H_{27}$	99494345	20.110	367.72	35.139

better for tree-like molecular enumeration, since all of these solutions are proceeded by keeping balance.

From Table 3.1, we can see that BfsSimEnum was faster than the other ones, which also implies that the employed and modified concepts of center rooted, left heavy and normal form are very useful for reducing the search space.

From the computational time shown in Table 3.2, we can see that BfsMulEnum is slightly less advantageous than Enumol when $\mathcal{M}_2 + 2\mathcal{M}_3 < 6$, which means that the number of double bonds is bounded by 5. As the number of multiple bonds increases (specially when $\mathcal{M}_2 + 2\mathcal{M}_3 \geq 6$), BfsMulEnum outperforms Enumol. The reason why BfsMulEnum is sometimes slower than Enumol is its dependence on BfsSimEnum. Due to this reason, there might be a large amount of simple trees computed by BfsSimEnum that cannot be expanded to multi-trees. On the other hand, our multi-tree enumeration method is significantly faster than Molgen.

Table 3.2: Comparison of BfsMulEnum with existing methods.

Molecular formula	$\mathcal{M}_2 + 2\mathcal{M}_3$	# Enumerated results	Computational time (Sec.)		
			BfsMulEnum	Molgen	Enumol
$C_{18}H_{34}$	2	3218346	0.266	145.99	0.311
$C_{19}H_{34}$	3	31503100	2.727	3753.04	2.7
$C_{20}H_{34}$	4	250132215	23.689	98799.2	23.39
$C_{20}H_{28}$	6	1185277179	181.37	-	188.6
$C_{22}H_{36}$	5	5445565067	556.21	-	544.52
$C_{22}H_{34}$	6	10198151506	1185.27	-	1192.53
$C_{22}H_{30}$	8	19663780677	3255.08	-	3392.54
$C_{10}O_4H_{16}$	3	10003272	1.5	400.83	1.335
$C_{12}O_4H_{16}$	5	282338151	63.33	176186.1	56.352
$C_{12}O_4H_{10}$	8	49498872	78.91	1183717.4	90.82
$C_{16}O_2H_{20}$	7	1996919931	467.48	-	470.21
$C_{16}O_4H_{30}$	2	12880695359	1172.81	-	1137.07
$C_6N_2O_3H_{14}$	1	643197	0.1	5.13	0.1
$C_6N_2O_3H_{10}$	3	1499019	0.345	44.01	0.307
$C_7N_2O_2H_{10}$	4	1312737	0.360	83.95	0.317
$C_7N_2O_2H_6$	6	257531	0.380	329.75	0.41
$C_7N_3O_2H_9$	5	8360420	3.932	1855.59	3.836
$C_7N_3O_2H_7$	6	3282844	3.81	11166.89	4.21
$C_8N_3O_2H_{11}$	5	62066528	20.931	16791.67	20.141
$C_8N_3O_2H_9$	6	31421502	21.52	70591.54	23.36
$C_9N_2O_2H_{10}$	6	18780376	10.038	15779.98	10.478
$C_9N_2O_2H_8$	7	7205103	9.27	76774.18	11.33
$C_9N_3O_2H_{11}$	6	252761084	119.06	288470.42	123.68
$C_9N_3O_2H_9$	7	107205329	113.67	637866.1	138.33
$C_{10}N_3O_2H_{11}$	7	932854039	637.2	-	715.99
$C_{11}N_3O_2H_{21}$	3	7268812476	802.67	-	774.56
$C_{11}N_3OH_{15}$	6	956851032	247.52	-	250.02

3.4.2 Extension to multivalent elements

We extended our algorithms to deal with multivalent elements. Our methods allow one element occur with different valences in a generated molecular tree. Since every atomic label in Σ has a prescribed valence, we set n distinct labels to represent atom with n multiple valences such that each of these labels is treated as an independent atom. For example, we set C and $C^{(2)}$ (whose valences are 4 and 2) to both represent carbon atom, while as defined in Σ, they are handled as two different labels with representing two different atoms. It is noted that this expansion requires the number of such multivalent elements being given in advance. However, we can remove this assumption by making exhaustive search on the numbers of C and $C^{(2)}$. Since the number of such combinations is bounded by the number of carbon atoms, it does not significantly increase the computational time. Although multivalent elements are not common for carbon atoms, they are common for sulfur and phosphorus atoms.

We also performed some experiments to verify this extension. Table 3.3 gives the number of generated trees and computational time. Since Molgen cannot deal

Table 3.3: Results for compounds with multivalent elements.

Molecular formula	# Enumerated results	Computational time (Sec.)	
		BfsSimEnum	Enumol
$C_{10}C_2^{(2)}O_2H_{22}$	1754152	0.39	0.6
$C_{16}C^{(2)}O_3H_{34}$	1068503824	228.8	403.16
$C_8C_2^{(2)}N_3O_2H_{21}$	602536450	122.82	229.21
$C_{11}C^{(2)}N_3OH_{27}$	195396579	40.45	72.11
$C_{10}C_2^{(2)}O_2H_{20}$	10917613	1.56	1.42
$C_{10}C^{(2)}O_2H_6$	288867	1.42	1.97
$C_{12}C_2^{(2)}O_2H_{10}$	289235948	444.81	517.13
$C_9C^{(2)}N_3OH_9$	212688117	221.31	270.83

with such extension, we only compared our generated results with that of Enumol. From the results, we can see that our extension is also correct and competitively faster than Enumol.

3.4.3 Structural matching

We performed some other experiments that examine whether each generated structure is found in the database. Here we used PubChem[1] to process the matching experiments. The matching rates of structures generated by both BfsSimEnum and BfsMulEnum are shown in Table 3.4 and Table 3.5, respectively. Since the valence of Hydrogen atom is one which can be spontaneously added as a leaf at last, the number of enumerated isomers is not affected by the number of H atoms, but is strongly relevant to the number of atoms whose valences are greater than 1. From these numbers of enumerated isomers, we can clearly see that the number of enumerated structures increases with the number of atoms with valence greater than 1 growing. From the mathcing results, we can also see that with increase of the number of generated structures, the matching ratio decreases. It indicates that a large amount of structures are exponentially generated, but only a small fraction of them has been explored. This finding is encouraging such that these unexplored regions provide a vast potentiality to improve our today's drugs by new compound design. Therefore, exploration of these unknown structures might be a crucial extension topic so as to discover new useful compounds, which will advance deep understanding of biology and lead to a new strategy to treat disease.

[1]http://pubchem.ncbi.nlm.nih.gov

Table 3.4: Matching results for chemical compounds only with single bonds to PubChem.

Molecular formula	# Enumerated results	Matching to PubChem	
		# Matched isomers	Existence rate
$C_2O_2H_6$	5	5	1.0
$C_3O_2H_8$	11	11	1.0
$C_3O_3H_8$	28	22	0.786
$C_4O_2H_{10}$	28	28	1.0
$C_4O_3H_{10}$	88	59	0.670
$C_4O_4H_{10}$	255	35	0.137
$C_5O_2H_{12}$	69	68	0.986
$C_5O_3H_{12}$	258	96	0.372
$C_5O_4H_{12}$	869	60	0.069
$C_5O_5H_{12}$	2570	8	0.003
$C_6O_2H_{14}$	179	147	0.821
$C_6O_3H_{14}$	772	177	0.229
$C_7O_2H_{16}$	463	282	0.609
$C_7O_3H_{16}$	2275	206	0.091
C_8OH_{18}	171	142	0.830
$C_8O_2H_{18}$	1225	407	0.332
C_9OH_{20}	405	221	0.546
$C_9O_2H_{20}$	3246	407	0.125
$C_{10}OH_{22}$	989	254	0.257

Further combination with chemical activity and biological property can help researchers to find useful molecules so as to solve real world problems such as molecular design and drug design, etc. For instance, let each element be weighted by its interacting ability with others so that we can predict each enumerated structure with a score. Using such scores, we can further predict new compounds and also reduce unreal isomers from the large unexplored chemical space. While that, BfsSimEnum and BfsMulEnum as rudimentary approaches provided an efficient route toward further combinations and applications.

3.5 Conclusion

In this study, we proposed BfsSimEnum and BfsMulEnum for enumerating tree-like compounds by firstly utilizing the breadth-first search order. Owing to the utilization of BFS order, both BfsSimEnum and BfsMulEnum only produce balanced intermediate trees during their search of a family tree without proceeding or stor-

Table 3.5: Matching results for chemical compounds with multiple bonds to PubChem.

Molecular formula	# Enumerated results	Matching to PubChem	
		# Matched isomers	Existence rate
$C_2O_2H_2$	4	4	1.0
$C_3O_2H_4$	19	14	0.737
$C_3O_2H_6$	19	15	0.789
$C_3O_3H_4$	43	10	0.233
$C_4O_2H_6$	80	32	0.4
$C_4O_2H_8$	66	44	0.667
$C_4O_3H_6$	241	43	0.178
$C_4O_3H_8$	212	66	0.311
$C_5O_2H_8$	308	87	0.282
$C_5O_2H_{10}$	204	102	0.5
$C_5O_3H_{10}$	798	137	0.172
$C_5O_4H_{10}$	2628	153	0.058
$C_6O_2H_{10}$	1139	169	0.148
$C_6O_2H_{12}$	641	206	0.321
$C_6O_3H_{12}$	2845	279	0.098
$C_7O_2H_{14}$	1946	277	0.142
$C_7O_3H_{14}$	9823	423	0.043
C_8OH_{14}	1863	247	0.133
C_8OH_{16}	790	228	0.289
C_9OH_{18}	2136	246	0.115
C_9OH_{16}	5626	231	0.041

ing any unbalanced ones, which can efficiently avoid duplicates. Together with the employed and modified concepts such as center-rooted, left-heavy and normal form, our proposed methods are successfully showed to be useful for reducing the large search space.

The results of computational experiments indicate that our algorithms are exact and faster than state-of-the-art ones for simple tree enumeration. But for multi-trees enumeration, BfsMulEnum is often outperformed by Enumol only when $\mathcal{M}_2 + 2\mathcal{M}_3$ is bounded to 5. Although it is efficient for molecules which include large number of multiple bonds ($\mathcal{M}_2 + 2\mathcal{M}_3 \geq 6$), BfsMulEnum is possible to get a further extension to make it independent from BfsSimEnum. For this purpose, not only possible vertices but also possible multi-edges should be both taken into account when generating intermediate trees. Such an extension can significantly reduce search space to speed up BfsMulEnum because it aims to generate intermediate trees without expanding simple trees which no longer can be changed to multi-trees. To let our proposed methods be more practical and widely enployed, improvement of BfsMulEnum will be our imperative next future step.

By finding the generated structures in the PubChem database, we obtained low matching rates which suggest that a large amount of structures are generated with representing unknown compounds. These unexplored structures are expected to carry important chemical characteristics lack of which may cause dysfunction or

diseases. Understanding of such unknown structures should be an essential future step so as to discover new compounds or/and even to design new drugs for diseases. We consider that combination of atomic chemical activity and biological property with these unknown structures should be useful to predict their potential functions.

BfsSimEnum and BfsMulEnum can also be extended to deal with some cyclic compounds such as benzenes by representing these cyclic structures as special vertices and by considering the symmetry. For example, benzene can be represented as an atom with valence 6, together with further restrictions such as giving special constraints on the ordering of 6 positions, or only single bonds are allowed to link to such an atom. Such an extension has already been implemented in Enumol and is also under development for BfsSimEnum and BfsMulEnum. Further extensions to include more complex ring structures such as Naphthalenes are also under development. Therefore, the methodologies developed here are not limited to enumeration of tree-like chemical compounds but are useful to cover a wide range of chemical compounds.

Our proposed methods are fast and fundamental for molecular enumeration that has many useful applications. Extensions toward enumerating general compounds and combination with biological properties should be interesting future work.

Chapter 4

Protein complex prediction via improved verification methods

4.1 Background

Protein-protein interactions (PPIs) in a living cell govern the cytoskeletal structures and molecular processes at both cellular and systemic levels, which refer to physical and functional contacts between two or more proteins. Since intensive studies have been carried out to examine PPIs in living systems in a few decades, enormous amounts of PPI data are available to understand important principles of cellular organization and biological function in cell biology and system biology. Protein complexes are known as clusters of multiple proteins linked by non-covalent physical protein-protein interactions that generally correspond to dense regions within PPI networks, which have been demonstrated to carry important cellular function in living cells. Although PPI data grows up rapidly, identification of protein complexes within PPI networks is still a vital topic due to the lack of known protein complexes.

Recent achievements have been proposed to identify protein complexes as dense subgraphs in PPI networks since proteins in a complex highly interact with each other. As novel graph theoretic clustering approaches, the MCL and MCODE algorithms have been successfully applied to cluster member proteins which often represent complexes within large PPI networks by using precomputed similarity information [23] and detecting connected regions [8], respectively. In addition, some other approaches such as SPC (superparamagnetic clustering), MC, and PCP are also proposed to use topological properties such that proteins in each complex are densely connected [17, 80]. Moreover, RNSC clustering algorithm uses a cost function to efficiently categorize member proteins from PPI networks [50]. [66] proposed a probabilistic algorithm to detect new protein complexes by training topological patterns and biological properties from known ones. [57] proposed NWE (Node-

Weighted Expansion of clusters of proteins) by introducing a random walk with restarts with a cluster of proteins. [91] used tandem affinity purification (TAP) data that is obtained from an experimental method for detecting multi-protein interactions.

However, the drawback of current methods is that they detect the dense regions as protein complexes without taking into account of structural constraints of proteins and topology of PPIs in the networks, which lead to a group of proteins incorrectly identified in extracted complexes. To date, some computational methods have been proposed to verify protein complexes by assessing the validation of individual interaction based on the topology of PPI networks. A verification algorithm AlternativePathFinder has been proposed to consider a complex candidate as a correct complex if it is included in a close loop [16]. One other achievement proposed by [32] detects protein complexes from the PPI networks based on finding maximal k-connected subgraphs. In addition, a clustering-based method, CMC, maximizes cliques from the weighted PPI networks to detect protein complexes. Unfortunately, these existing methods still have low precision since they did not consider structural constraints of proteins in PPI networks. Ozawa et al. has proposed an integer programming-based method that verify and reconstruct the topology of domain-domain interactions in PPI networks [63]. Conceptually, the basic idea of this approach is that proteins in complex candidates, each of whose domains participate only in a single interaction, can form a valid protein complex. And their method seeks for optimal combinations of domain-domain interactions (DDIs) in the complex candidates predicted from other existing methods. They proposed the optimization problem to extract subgraphs from complex candidates that contain more than one proteins connected by more than one DDI as verified protein complexes. Although this approach has achieved a relatively high precision, it still outputs a number of false positives and requires further improvement.

In this study, we improve the method by Ozawa et al. and propose a novel formulation of integer programming for verifying candidate complexes predicted by existing graph clustering methods. Our method is based on the idea that a candidate complex should not be divided into many small complexes. In addition, we use maximal components and extreme sets that are defined based on edge connectivity in graph theory [60]. Since the internal proteins of a maximal component are connected more strongly with each other than with any other external proteins, as well as an extreme set, they are expected to be useful to further increase the precision. Furthermore, we prove that the problem of maximizing the number of protein-protein interactions, considered by Ozawa et al., is NP-hard, and also prove that our problem of maximizing the size of a connected component given by verified protein-protein interactions is NP-hard. Finally, we apply this improved IP-based method and its combination methods with maximal components and extreme sets to some computational experiments.

4.2 Methods

An integer programming (IP)-based method has been developed by Ozawa et al. to verify candidate complexes by maximizing the number of protein-protein interactions [63]. In this section, we propose a novel formulation of integer programming based on the idea that a candidate complex should not be divided into many small complexes, and improve their method. In addition, we propose combinations of the improved method with maximal components and extreme sets [60].

4.2.1 Improved integer programming IPc

Ozawa et al. proposed their IP-based method by verifying an interaction between two proteins depending on the presence of interactions between domains included in the proteins. A domain is assumed to interact with at most one other domain, which means that if a domain can interact with multiple domains, only one domain is selected as its partner. In their original IP-based method, such pairs of domains are selected by maximizing the number of interacting protein pairs. However, candidate proteins should be connected as much as possible because the proteins are selected as a complex by prediction methods such as MCODE, MCL, and RNSC. Therefore, based on the concept that a domain interacts with at most one domain, we consider the problem of finding the largest set of proteins that are connected to each other. Herein, we call the original IP proposed by Ozawa et al. and our improved IP, 'IPo' and 'IPc' for short, respectively.

Let $\mathcal{P}$ and $\mathcal{D}$ be a set of candidate proteins for constituting a complex and a set of domains included in the proteins of $\mathcal{P}$, respectively, where each domain $i.k \in \mathcal{D}$ is distinguished by the protein i that the domain k belongs to. Let $\mathcal{I}_{\mathcal{P}}$, $\mathcal{I}_{\mathcal{D}}$, and $\mathcal{I}_{\mathcal{D}i,j}$ be a set of potentially interacting protein pairs, a set of potentially interacting domain pairs, and a set of potentially interacting domain pairs between proteins i and j, respectively. We approximate the problem of maximizing the size of a connected component of proteins into that of maximizing the number of connected components with size three. And this approximated problem can be simply transformed into the following integer program.

$$
\begin{aligned}
&\text{Maximize } \textstyle\sum_{i,j,k\in\mathcal{P}} x_{i,j,k},\\
&\text{Subject to}\\
&\sum_{\{(i.k,j.l)\in\mathcal{I}_{\mathcal{D}} | i.k=m.n \text{ or } j.l=m.n\}} d_{i.k,j.l} \le 1 \qquad \text{for all } m.n \in \mathcal{D}, \qquad (1)\\
&p_{i,j} \le \sum_{(i.k,j.l)\in\mathcal{I}_{\mathcal{D}i,j}} d_{i.k,j.l} \qquad \text{for all } (i,j) \in \mathcal{I}_{\mathcal{P}}, \qquad (2)\\
&x_{i,j,k} \le \frac{1}{2}(p_{i,j} + p_{j,k} + p_{i,k}) \qquad \text{for all } i,j,k \in \mathcal{P}. \qquad (3)
\end{aligned}
$$

In the above inequalities, each variable of $x_{i,j,k}$, $p_{i,j}$, and $d_{i.k,j.l}$ takes 0 or 1. $x_{i,j,k} = 1$ if and only if proteins i, j, and k are connected. $p_{i,j} = 1$ if and only if proteins i and j interact with each other. $d_{i.k,j.l} = 1$ if and only if domains $i.k$ and $j.l$ interact with each other. It should be noted that for variables $x_{i,j,k}$, we do not need to treat all combinations of three proteins, but only need to investigate that proteins can be connected. Thus, the number of variables $x_{i,j,k}$ is at most $\begin{pmatrix} |\mathcal{I}_{\mathcal{P}}| \\ 2 \end{pmatrix}$. The inequalities (1) and (2) are also included in IPo by Ozawa et al. Same with IPo, we assume that each complex consists of at least three proteins. It should be noted that the topology of protein-protein interaction networks is taken into account in Eq. (2). And the meaning of each inequality is as follows:

(1) The number of domains that interact with domain $m.n$ is at most one.

(2) Proteins i and j interact if and only if there is at least one interacting domain pair $(i.k, j.l)$.

(3) Proteins i, j, and k are connected if and only if there are at least two interacting protein pairs from (i, j), (j, k), and (i, k).

An illustration of verification by these IP-based methods is given in Figure 4.1 . Figure 4.1(a) shows an example of a protein interaction network and domain-domain interactions. There are six proteins $P_1, \ldots, P_6$ that contain one or two domains, $\{D_1\}$, $\{D_2, D_3\}$, $\{D_4, D_5\}$, $\{D_6\}$, $\{D_7, D_8\}$ and $\{D_9, D_{10}\}$, respectively. There are seven potentially interacting domain pairs $\mathcal{I}_{\mathcal{D}} = \{(D_1, D_7), (D_2, D_7), (D_2, D_9), (D_3, D_4), (D_5, D_{10}), (D_6, D_8), (D_8, D_9)\}$, and seven potentially interacting protein pairs $\mathcal{I}_{\mathcal{P}} = \{(P_1, P_5), (P_2, P_5), (P_2, P_6), (P_2, P_3), (P_3, P_6), (P_4, P_5), (P_5, P_6)\}$. Figure 4.1(b) shows the optimal solution of this example by IPo. A candidate complex is divided into two complexes $\{P_1, P_4, P_5\}$ and $\{P_2, P_3, P_6\}$. The value of the objective function of IPo, that is, the maximum number of verified interacting protein pairs is 5. Figure 4.1(c) shows the optimal solution by IPc, where only one protein complex $\{P_2, P_3, P_5, P_6\}$ is generated with value of objective function being 4. Although the optimal score of IPo is better than that of IPc, IPc outputs more reasonable results than IPo since a larger cluster remains in the solution by IPc.

4.2.2 Maximal components and extreme sets

In this subsection, we use maximal components and extreme sets in graph theory to enhance the verification ability of IPc method, where both of these two concepts are defined by using edge connectivity.

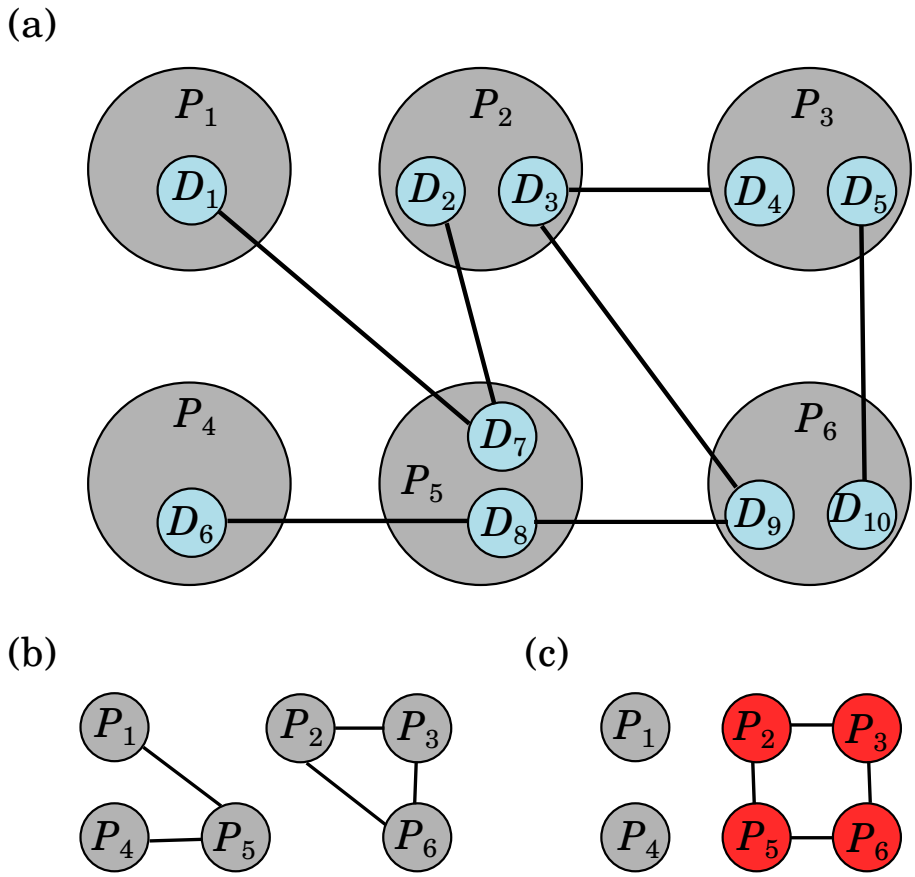


Figure 4.1: Example of verification by two IP-based methods, IPo and IPc. (a) Example of a protein interaction network and domain-domain interactions. There are six proteins that contain one or two domains, seven potentially interacting domain pairs, and seven potentially interacting protein pairs, where these protein-protein interactions are not shown. (b) The optimal solution by the IP of [63], IPo. Each solid line denotes a protein-protein interaction. Two protein complexes are generated. (c) The optimal solution by our proposed IP, IPc. A larger protein complex is generated.

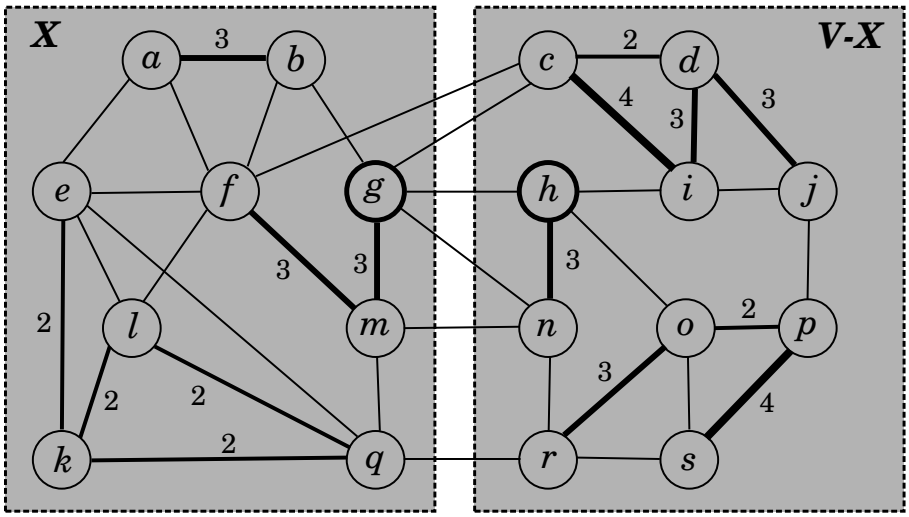


Figure 4.2: Illustration of a cut $\{X, V - X\}$ that determines the local edge-connectivity $\lambda_G(g, h)$ between vertices g and h, where the graph G contains the set of 19 vertices $V = \{a, b, \ldots, s\}$ and the set of edges E, each number in this figure denotes the weight w_G of the edge, and the edges without a number are weighted by 1. For the set $X = \{a, b, e, f, g, l, m, k, q\}$, $\sum_{u \in X, v \in V-X} w_G(u, v) = 6$. Then, X gives one of the minimum (g, h)-cuts in Figure 4.3, and $\lambda_G(g, h) = 6$.

Local edge-connectivity

Let $G(V, E)$ be an undirected edge-weighted graph with a set of vertices V and a set of edges E. Each edge e has a non-negative real weight $w_G(e)$. The *local edge-connectivity* $\lambda_G(u, v)$ between two nodes u and v is defined as follows [60].

$$\lambda_G(u, v) \quad = \quad \min_{\{X \subset V | u \in X, v \in V - X\}} d_G(X),$$

where $d_G(X)$ denotes the cut size of $\{X, V-X\}$, formulated by $\sum_{u \in X, v \in V-X} w_G(u, v)$ (see also an example in Figure 4.2). It is to be noted that if the local edge-connectivity $\lambda_G(u, v)$ between u and v is large, then the relationship between them is strong.

Maximal component

A subset X of V is called a *maximal component* of a graph G if it satisfies the following conditions,

$$\begin{aligned} &\lambda_G(u, v) \geq l && \text{for } \forall u, v \in X, \\ &\lambda_G(u, v) < l && \text{for } \forall u \in X, \ \forall v \in V - X, \end{aligned}$$

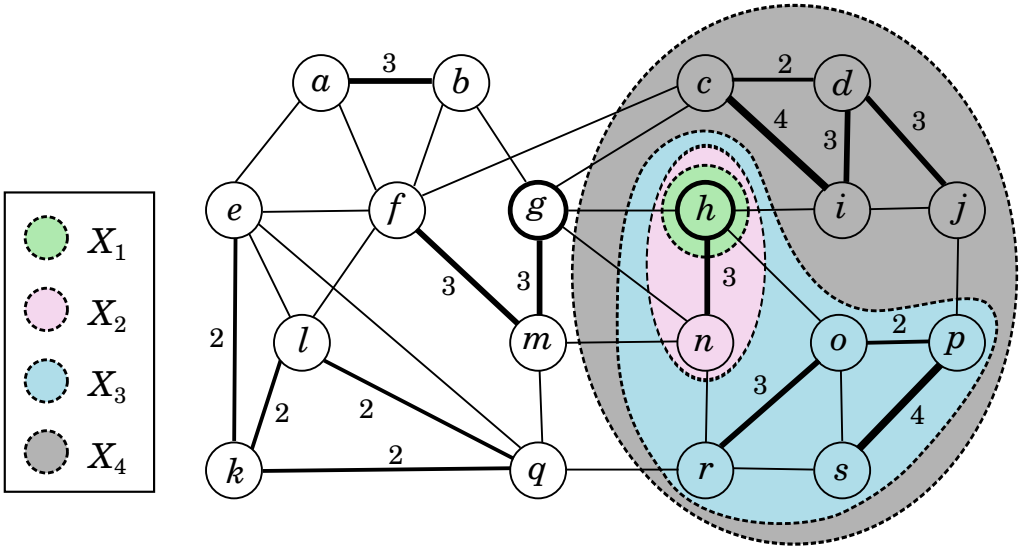


Figure 4.3: Minimum (g, h)-cuts of the graph G in Figure 4.2. There are four cuts given by the sets including the vertex h, $X_1 = \{h\}$, $X_2 = \{h, n\}$, $X_3 = \{h, n, o, p, r, s\}$, and $X_4 = \{c, d, h, i, j, n, o, p, r, s\}$.

where $l = \min_{u,v \in X} \lambda_G(u, v)$. It means that the internal vertices of a maximal component are connected more strongly with each other than with any other external vertices.

Extreme set

A nonempty proper subset X of V is called an *extreme set* of a graph G if it satisfies the following condition,

$$d_G(X) < d_G(Y) \qquad \text{for } \forall\, Y \subset X.$$

It is known that every extreme set is a maximal component, and there exists an $O(mn + n^2 \log n)$ time algorithm that computes maximal components and extreme sets of a graph with n vertices and m edges [60].

Figure 4.4 shows the maximal components and the extreme sets for the graph of Figure 4.2. Each colored area corresponds to a maximal component (an extreme set and a maximal component).

For verifying protein complexes, we let $w_G(u, v) = 1$ for each protein-protein interaction, and calculate maximal components and extreme sets.

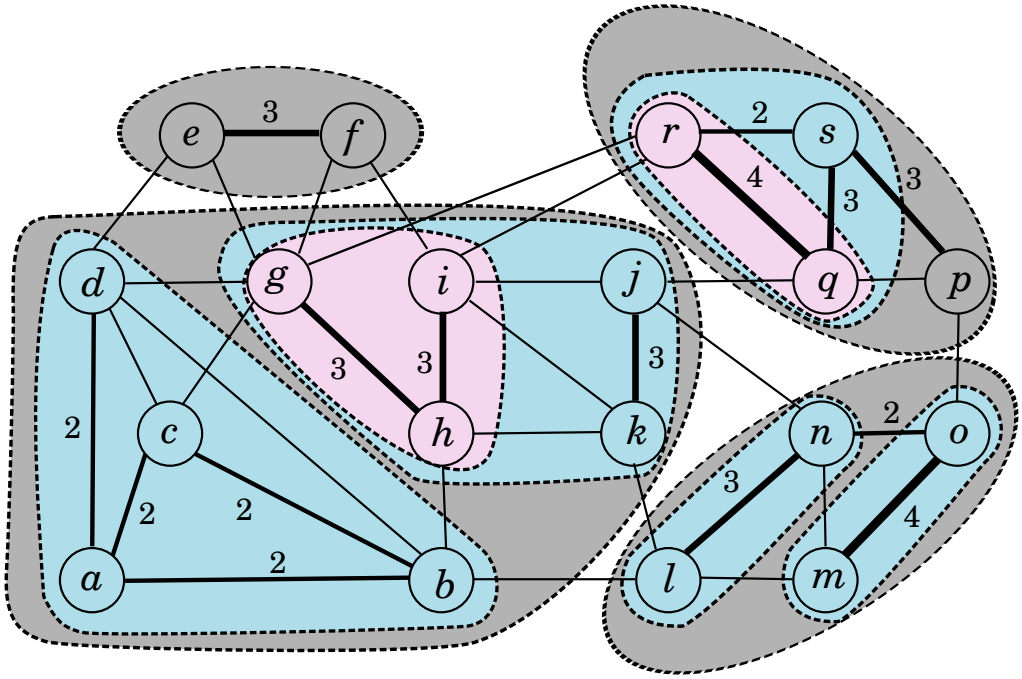


Figure 4.4: Illustration of maximal components and extreme sets. The maximal components and the extreme sets of the graph G in Figure 4.2. Each colored area corresponds to a maximal component (an extreme set and a maximal component).

4.3 Hardness analysis

In this section, we define the problem formulated as an integer program by Ozawa et al. and prove that it is NP-hard. In addition, we also prove that the problem of maximizing the size of a connected component in a protein complex is NP-hard.

Problem 1. *Protein Complex Verification Problem (PCVP)*

Given a set of proteins $\mathcal{P}$ as a candidate complex, a set of domains $\mathcal{D}$ included in the proteins, a set of potential protein-protein interactions $\mathcal{I}_{\mathcal{P}}$, and a set of potential domain-domain interactions $\mathcal{I}_{\mathcal{D}}$, maximize the number of verified protein-protein interactions, where a domain can interact with at most one domain, and a protein-protein interaction is said to be verified if the corresponding pair of proteins contains at least one interacting domain pair.

Then, we have the following theorem.

Theorem 4.3.1. *The protein complex verification problem (PCVP) is NP-hard.*

Proof. We present a polynomial-time reduction from 3-dimensional matching.

Problem 2. *3-dimensional matching (3DM)*

Given $\mathcal{S} \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, where $\mathcal{X}, \mathcal{Y}$, and $\mathcal{Z}$ are finite mutually disjoint sets with $|\mathcal{X}| = |\mathcal{Y}| = |\mathcal{Z}| = n$, find $\mathcal{M} \subseteq \mathcal{S}$ such that $|\mathcal{M}| = n$ and $\{X, Y, Z \mid (X, Y, Z) \in \mathcal{M}\} = \mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$.

Suppose that an instance of 3DM is given as $\mathcal{S} \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, where $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$, $\mathcal{Y} = \{Y_1, Y_2, \ldots, Y_n\}$ and $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_n\}$ are three mutually disjoint sets with n elements, respectively. We then transform the instance of 3DM to an instance of PCVP as follows (see also an illustration in Figure 4.5).

$$
\begin{aligned}
\mathcal{P} &= \{\mathcal{P}_{\mathcal{X}}\} \cup \bigcup_{j=1}^{n} \{\mathcal{P}_{Y_j}\} \cup \bigcup_{k=1}^{n} \bigcup_{l=1}^{n-1} \{\mathcal{P}_{Z_k}^{(l)}\}, \\
\mathcal{D} &= \bigcup_{i=1}^{n} \{X_i\} \cup \bigcup_{j=1}^{n} \bigcup_{k=1}^{n} \{D_{Y_j}^{Z_k}\} \cup \bigcup_{k=1}^{n} \bigcup_{l=1}^{n-1} \{Z_k^{(l)}\}, \\
\mathcal{I}_{\mathcal{D}} &= \{(X_i, D_{Y_j}^{Z_k}) \mid (X_i, Y_j, Z_k) \in \mathcal{S}\} \cup \bigcup_{j=1}^{n} \bigcup_{k=1}^{n} \bigcup_{l=1}^{n-1} \{(D_{Y_j}^{Z_k}, Z_k^{(l)})\},
\end{aligned}
$$

where $\mathcal{P}_{\mathcal{X}}$ contains domains $X_1, X_2, \ldots$, and X_n ($\in \mathcal{X}$), $\mathcal{P}_{Y_j}$ ($Y_j \in \mathcal{Y}$) contains domains $D_{Y_j}^{Z_1}, D_{Y_j}^{Z_2}, \ldots$, and $D_{Y_j}^{Z_n}$, $\mathcal{P}_{Z_k}^{(l)}$ ($Z_k \in \mathcal{Z}, l \in \{1, \ldots, n-1\}$) contains a domain $Z_k^{(l)}$. If $(\alpha, \beta) \in \mathcal{I}_{\mathcal{D}}$ and α and β are respectively included in proteins γ and δ, then we let $(\gamma, \delta) \in \mathcal{I}_{\mathcal{P}}$.

Then, we can see that there exists a 3-dimensional matching if and only if the maximum number of verified protein-protein interactions is exactly $(n + n \cdot (n-1)) = n^2$, and $(X_i, Y_j, Z_k) \in \mathcal{M}$ holds if and only if $(X_i, D_{Y_j}^{Z_k})$ is selected. For each protein pair $(\mathcal{P}_{\mathcal{X}}, \mathcal{P}_{Y_j})$, exactly one domain pair is selected because protein $\mathcal{P}_{\mathcal{X}}$ contains n domains, and a domain interacts with at most one domain. If more than one domain pair are selected for a protein pair, the number of proteins that interact with $\mathcal{P}_{\mathcal{X}}$ is less than n, that is, the maximum cannot achieve n^2. Furthermore, among n domains $D_{Y_j}^{Z_k}$ for each Z_k ($\in \mathcal{Z}$), exactly one domain $D_{Y_j}^{Z_k}$ is selected for the interaction with $\mathcal{P}_{\mathcal{X}}$, and the other domains interact with $(n-1)$ domains of $Z_k^{(l)}$ ($l = 1, \ldots, n-1$), respectively. If more than one domain are selected for the interaction with $\mathcal{P}_{\mathcal{X}}$, there exists the protein $\mathcal{P}_{Z_k}^{(l)}$ for some $l \in \{1, \ldots, n-1\}$ that is not able to interact with any protein.

Furthermore, we can derive a solution for an instance of 3DM in polynomial time from a solution for the transformed instance of PCVP. An instance of 3DM can also be transformed in polynomial time to the corresponding instance of PCVP. Therefore, the protein complex verification problem (PCVP) is NP-hard. □

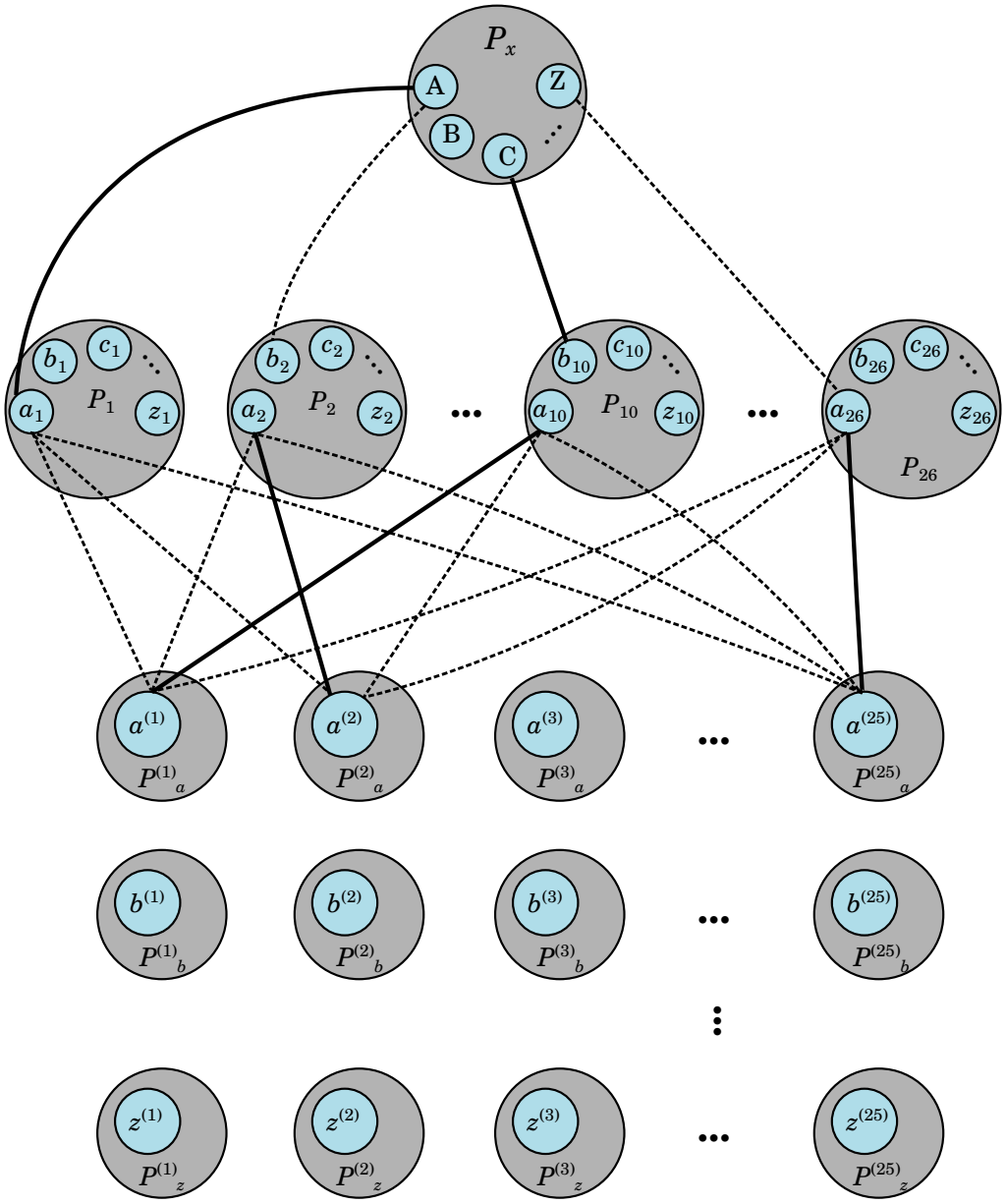


Figure 4.5: Illustration of the reduction from 3-dimensional matching (3DM) to the protein complex verification problem (PCVP) in the case of $n = 26$, where $\mathcal{X} = \{A, B, \ldots, Z\}$, $\mathcal{Y} = \{1, 2, \ldots, 26\}$, and $\mathcal{Z} = \{a, b, \ldots, z\}$. The large and small circles denote proteins and domains, respectively. The solid and dotted lines denote potential domain-domain interactions, and the solid lines are selected.

Similarly, we can also prove that the problem of maximizing the size of a connected component consisting of proteins is NP-hard.

Problem 3. *Connected Protein Complex Verification Problem (CPCVP)*
Given a set of proteins $\mathcal{P}$ as a candidate complex, a set of domains $\mathcal{D}$ included in the proteins, a set of potential protein-protein interactions $\mathcal{I}_\mathcal{P}$, and a set of potential domain-domain interactions $\mathcal{I}_\mathcal{D}$, maximize the size of a connected component given by verified protein-protein interactions, where a domain can interact with at most one domain.

Theorem 4.3.2. *The connected protein complex verification problem (CPCVP) is NP-hard.*

Proof. We prove this theorem by using the same reduction as in the proof of Theorem 4.3.1. There exists a 3-dimensional matching if and only if all the proteins in $\mathcal{P}$ are connected, and $(X_i, Y_j, Z_k) \in \mathcal{M}$ holds if and only if $(X_i, D_{Y_j}^{Z_k})$ is selected. Furthermore, the derivation of a solution of 3DM and the transformation of an instance of 3DM can be done in polynomial time. Therefore, the connected protein complex verification problem (CPCVP) is NP-hard. □

It is to be noted that maximizing the number of domain-domain interactions (instead of protein-protein interactions) can be done in polynomial time by using maximum matching.

4.4 Results

To confirm the advantage of our method, we performed some computational experiments, and compared the results with that of the original IPo which is shown to be the best existing method for verifying protein complexes [13].

4.4.1 Data and implementation

For the experiments, we select to use yeast PPI data. We used WI-PHI [49] and BioGRID [83] as PPI data, which compose of 5,907 and 4,603 proteins identified by UniProt database (Release 2011_03) [18], and 49,847 and 30,853 interacting protein pairs, respectively. For each protein, we extracted Pfam domains [11] included in the protein using the UniProt database. We used iPfam database (version 21.0) [26] as DDI data, which includes 2,837 Pfam domains and 4,030 interacting Pfam domain pairs. We applied existing prediction methods MCL [23] and MCODE [8] with parameters of 'inflation' and 'node score cutoff', respectively, to obtain candidate protein complexes from both of the WI-PHI and BioGRID PPI data. We used

CYC2008 [65] as experimental known protein complexes, which includes 408 curated complexes to assess the performances of verification methods.

For a set of verified protein complexes $\mathcal{C}$ and a set of known protein complexes $\mathcal{K}$, we computed the precision and recall by the following formulations (which have also been used in [17, 63]):

$$\begin{aligned} precision &= \frac{|\{c \in \mathcal{C} | \exists k \in \mathcal{K}\ concordance(c,k) \geq 0.5\}|}{|\mathcal{C}|}, \\ recall &= \frac{|\{k \in \mathcal{K} | \exists c \in \mathcal{C}\ concordance(c,k) \geq 0.5\}|}{|\mathcal{K}|}, \end{aligned}$$

where $concordance(c,k)$ denotes the concordance rate between sets of proteins c and k, which is defined as $\frac{|c \cap k|}{\sqrt{|c| \cdot |k|}}$. It is to be noted that there exist multiple predicted complexes that may correspond to the same known complex. The *accuracy* is defined as the geometrical mean of the precision and the recall: $accuracy = \sqrt{precision \cdot recall}$.

We used IBM ILOG CPLEX (version 12.1) to solve the integer programs. We performed the computational experiments on a PC with a Xeon CPU 3.33 GHz and 10 GB memory under the linux OS (version 2.6.16).

4.4.2 Comparison with existing methods

We obtained candidate complexes by using both MCL [23] and MCODE [8] and compared the performances of IPc and IPo.

Figure 4.6 shows the results of the precision by the original IP-based method (IPo), our improved IP-based method (IPc), maximal components, extreme sets, and the combination methods of IPc with maximal components (maximal+IPc) and extreme sets (extreme+IPc) for candidate protein complexes obtained from the WI-PHI and BioGRID PPI data, respectively, by MCL with varying the inflation parameter from 1.5 to 2.5. In the combination methods, IPc is applied after the calculation of maximal components and extreme sets, respectively. Each method was applied to candidate protein complexes obtained by MCL. For the WI-PHI data, the precision of IPc was better than that of IPo except for inflation=2.3, and in almost all methods, the precision was the best for inflation=2.1. For the BioGRID data, the precision of IPc was better than or comparable to that of IPo, and among all methods, the precision of IPc for inflation=1.8 was the best.

Figure 4.7 shows the results of the precision by the original IP-based method (IPo), our improved method (IPc), maximal components, extreme sets, maximal+IPc, and extreme+IPc for candidate protein complexes obtained from the WI-PHI and BioGRID protein-protein interaction data, respectively, by MCODE with varying the inflation parameter from 0.0 to 0.3. For both protein-protein interaction data,

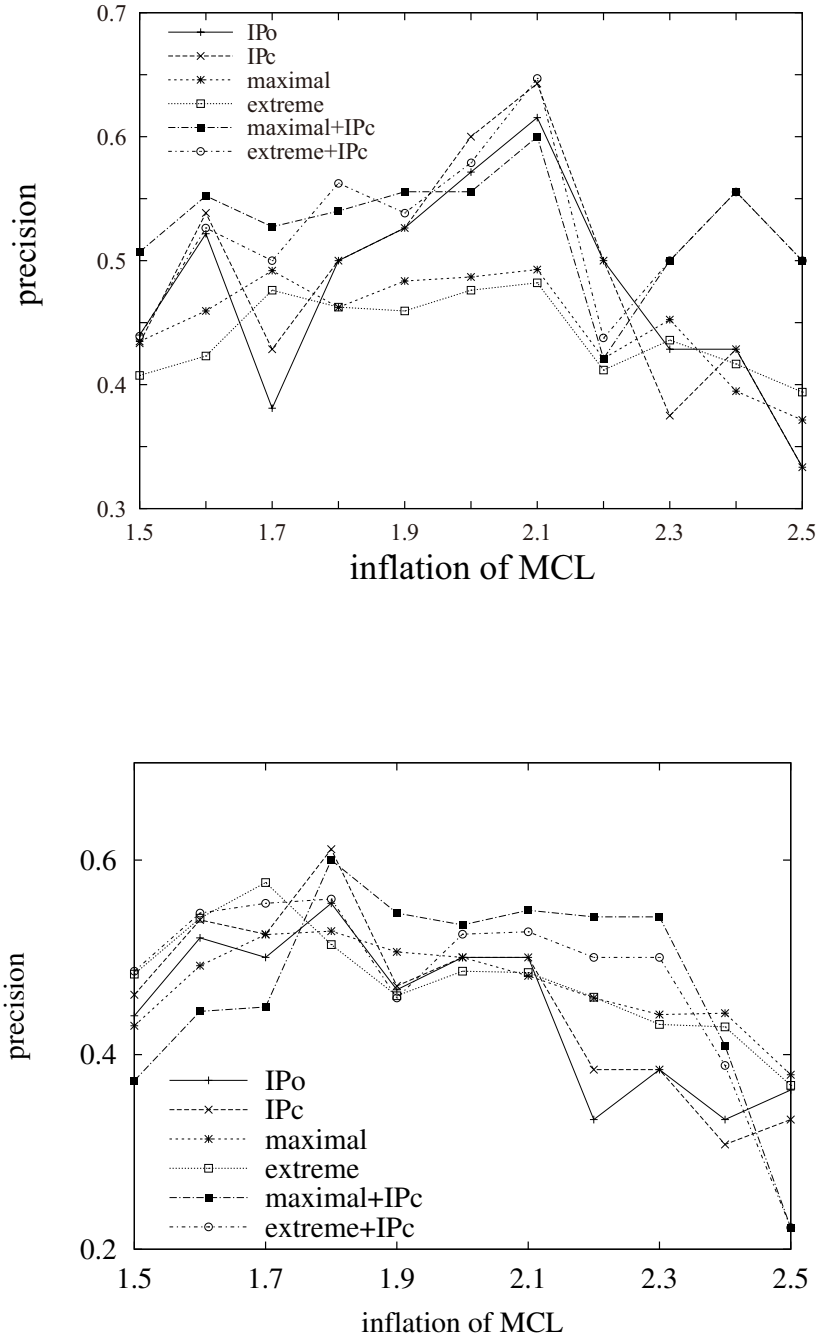


Figure 4.6: Results of the precision by IPo, IPc, maximal, extreme, maximal+IPc, and extreme+IPc for candidates obtained from WI-PHI (top) and BioGRID (bottom) by MCL with varying the inflation parameter from 1.5 to 2.5. 'maximal+IPc' and 'extreme+IPc' denote that IPc is applied after the calculation of maximal components and extreme sets, respectively. Each method was applied to candidate protein complexes.

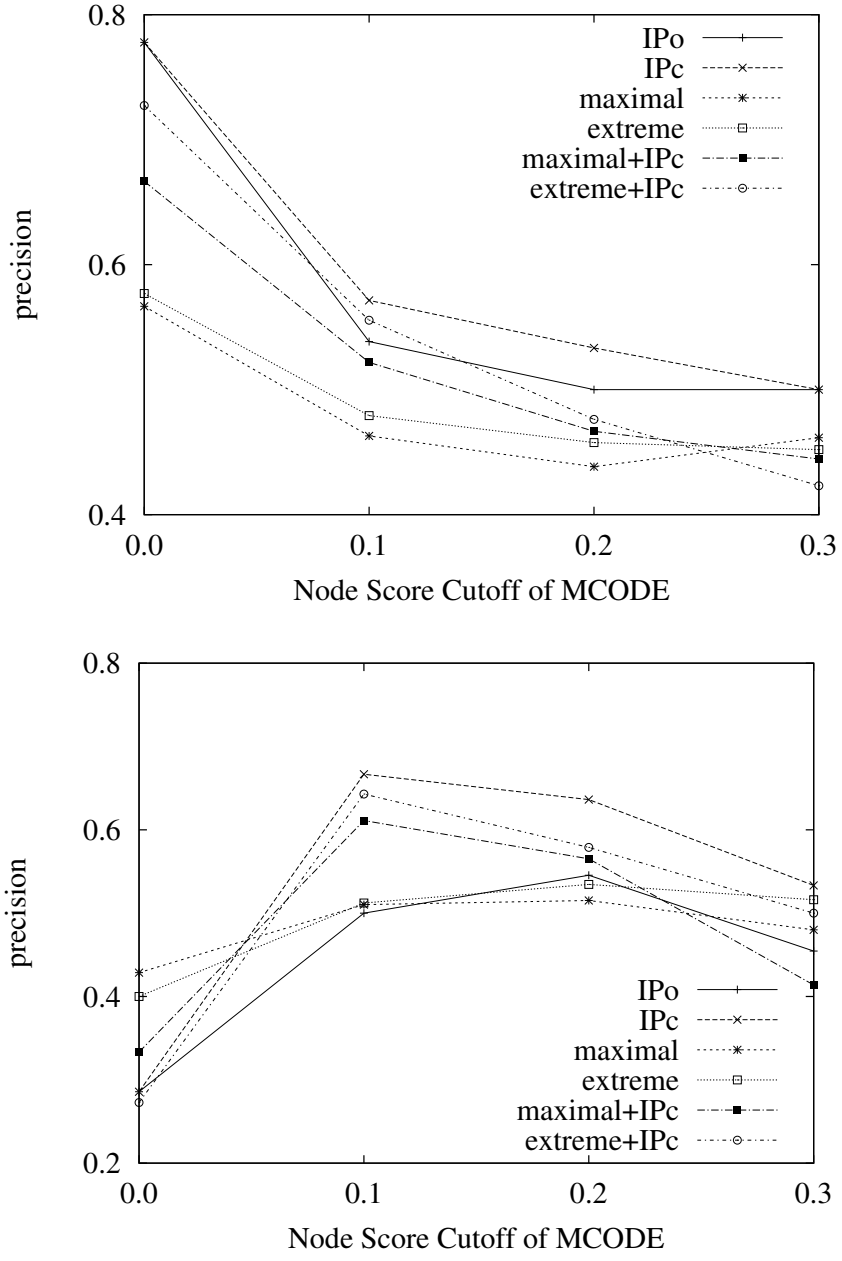


Figure 4.7: Results of the precision by IPo, IPc, maximal, extreme, maximal+IPc, and extreme+IPc for candidates obtained from WI-PHI (top) and BioGRID (bottom) by MCODE with varying the node score cutoff parameter from 0.0 to 0.3.

Table 4.1: Results of the precision, the recall, and the accuracy by the original IP-based method (IPo), our improved IP-based method (IPc), maximal components, extreme sets, maximal+IPc, and extreme+IPc for candidate protein complexes obtained from the WI-PHI data by MCL with inflation 1.9.

method	precision	recall	accuracy
IPo	0.526316	0.036765	0.139104
IPc	0.526316	0.036765	0.139104
maximal	0.483516	0.098039	0.217724
extreme	0.459459	0.090686	0.204124
maximal+IPc	0.555556	0.039216	0.147602
extreme+IPc	0.538462	0.036765	0.140700

Table 4.2: Results of the precision, the recall, and the accuracy by the original IP-based method (IPo), our improved IP-based method (IPc), maximal components, extreme sets, maximal+IPc, and extreme+IPc for candidate protein complexes obtained from the WI-PHI data by MCL with inflation 2.0.

method	precision	recall	accuracy
IPo	0.571429	0.031863	0.134934
IPc	0.600000	0.034314	0.143486
maximal	0.486842	0.093137	0.212939
extreme	0.476190	0.085784	0.202113
maximal+IPc	0.555556	0.036765	0.142915
extreme+IPc	0.578947	0.034314	0.140946

the precision of IPc was better than or comparable to that of IPo, and among all methods, the precision of IPc was the best except for the BioGRID data with cutoff=0.0.

Tables 4.1, 4.2, and 4.3 show the results of the precision, recall, and accuracy by IPo, IPc, maximal components, extreme sets, maximal+IPc, and extreme+IPc for candidate protein complexes obtained from the WI-PHI PPI data by MCL with inflation 1.9, 2.0, and 2.1, respectively. For inflation=1.9, the precision of IPo was equal to that of IPc and less than that of maximal+IPc and extreme+IPc, the precision of maximal+IPc was the best, and IPc output the same solution as IPo. For inflation=2.0, 2.1, the precisions of IPc and extreme+IPc were better than that of IPo. The recalls and accuracies of our methods were better than those of IPo, and the precision of extreme+IPc was the best when inflation=2.1. These results

Table 4.3: Results of the precision, the recall, and the accuracy by the original IP-based method (IPo), our improved IP-based method (IPc), maximal components, extreme sets, maximal+IPc, and extreme+IPc for candidate protein complexes obtained from the WI-PHI data by MCL with inflation 2.1.

method	precision	recall	accuracy
IPo	0.615385	0.031863	0.140028
IPc	0.642857	0.034314	0.148522
maximal	0.492754	0.088235	0.208514
extreme	0.482143	0.085784	0.203372
maximal+IPc	0.600000	0.036765	0.148522
extreme+IPc	0.647059	0.034314	0.149007

suggest that our proposed IP-based methods, especially extreme+IPc, considerably outperform IPo both in recall and precision. The maximum execution times of IPo and IPc for a candidate protein complex by MCL with inflation 2.1 were about 0.04 and 0.84 seconds, respectively, where both of the methods took less than 0.01 second per complex in most cases. Though IPc took longer CPU time than IPo did, it is still acceptable since the achievement of better precision of our method. Therefore, we can conclude that IPc is better than IPo.

4.5 Conclusion

We have addressed the problem of verification of candidate protein complexes, and proposed an improved integer programming (IP)-based method by maximizing the size of a connected component. In addition, we proposed the combination methods with maximal components and extreme sets, which partition vertices based on edge connectivity in graph theory. The results of comparison of the verification performance of our methods with existing IPo suggest that our proposed methods outperform the existing one.

Furthermore, we proved that the problem of maximizing the number of protein-protein interactions under the condition that a domain interacts with at most one other domain, considered by Ozawa et al., is NP-hard, and also proved that the problem of maximizing the size of a connected component considered by our method under the same condition is NP-hard as well. These results justify the use of integer programming both in our method and in the method by [63].

An interesting future step is to find a compact formulation of the problem of maximizing the size of a connected component since we solved this problem approx-

imately. New methods are also needed to achieve a better recall and to improve the efficiency factor to a higher range.

Chapter 5

Modeling the robustness of metabolic networks

5.1 Background

Metabolic pathways are complex systems of biochemical reactions taking place in every living cell to degrade substrates and synthesize molecules needed for life. Any metabolic dysfunction may lead to the impossibility to degrade or produce crucial molecules for the organism, potentially inducing disease or death. Yet cells seem to be able to maintain their normal functions despite many perturbations, such as the gene knock-out or DNA mutations perturbing the functions of proteins, while being very sensitive to some specific attacks [43]. Understanding and modeling the organizational principles underlying the robustness of metabolic networks with respect to gene perturbations is important not only to shed light on basic principles of life, but also to identify weaknesses which may lead to new drug targets to kill pathogens or cancer cells [12].

Conceptually, a metabolic network can be considered as a network consisting of metabolites and enzyme (gene)-catalyzed reactions which bridge these metabolites to transformation processes. A gene perturbation, such as knock-out or DNA mutation, can inhibit one or several reactions in a metabolic system. The impact of this perturbation on the cell phenotype can vary widely, ranging from no effect to cell death, depending on how many other reactions and crucial metabolites are impacted in cascade.

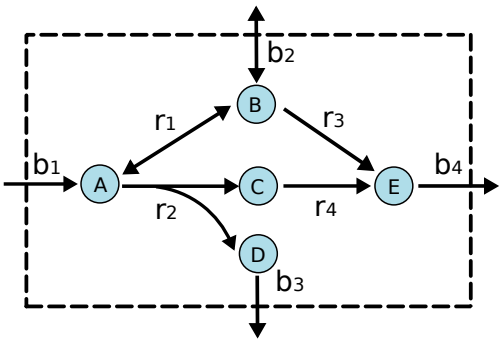
Several approaches have been proposed to model and predict the phenotypic impact of inhibiting one or several genes through metabolic network perturbation. Flux balance analysis (FBA) is a constraint-based mathematical model which uses the stoichiometry of a given metabolic network along with a biologically relevant objective function to identify optimal reaction flux distributions [67, 89]. It can be

used to predict the effect of inhibiting one or several reactions by assessing how the objective function changes after the perturbation [22]. A related approach proposed by [76] is the method of minimization of metabolic adjustment (MOMA), which predicts the flux vectors of gene knock-out mutants by imposing the constraint that mutants operate by minimizing their metabolic adjustment with respect to the wildtype. Flux variability analysis (FVA) assesses the range of possible fluxes for each reaction when the system runs near optimality, and has been used to evaluate the consequences of metabolic perturbation [78]; however FVA has not been used, to our knowledge, to predict metabolic gene essentiality. A limitation of FBA, MOMA and FVA is the difficulty to define a relevant objective function: for example, the objective function to predict cell growth typically involves a linear combination of more than 100 metabolites [67].

Other approaches model the effect of gene knock-out using the concept of elementary modes (EMs), which are minimal sets of reactions that can operate at the steady state, such that all irreversible reactions involved are used in the appropriate direction [73, 74]. Figure 5.1 shows for example the EMs of a simple network. With elementary mode analysis (EMA), [84] proposed that the viability of a mutant carrying mutation in a single gene can be predicted by the number of EMs which do not require the gene, a concept that has been generalized to define a notion of network robustness [12, 90]. EM-based methods, however, suffer from computational cost. Although several tools exist to compute EMs of middle-size networks [53, 88], they do not scale to large networks because the number of EMs grows exponentially with the network size [1, 36, 54]. [1] proved that counting the number of EMs is *#P-complete*, and although [36] proposed an efficient method for computing EMs, it is still not polynomial.

Alternatively, [52] proposed a model of minimal cut set (MCS) as a minimal set of reactions in metabolic networks whose disturbances cause dysfunction. However their computation of the MCSs is based on EMs, which becomes infeasible to analyze large-scale metabolic networks. A method based on a dual framework was recently proposed by [9], which can determine MCSs without calculating the EMs; the formulation is however also not scalable for large networks. Finally, different from many other approaches, the concept of synthetic accessibility (SA), proposed by [92], predicts the viability of mutant strains from the network topology, without knowledge of stoichiometry or biomass growth, but with specification of medium inputs and biomass outputs.

An alternative route to model the impact of a perturbation on a metabolic network is to start from a dynamic model of metabolism and assess how the model is impacted when a reaction is inhibited. Boolean models, in particular, are popular to describe and analyze large-scale metabolic networks [33, 82, 87]. Concepts of damage [55, 79] and topological impact degree [45] were extensively studied in recent years, where [55, 79] and [45] define the impact of a reaction as the number of



Elementary modes

	r1	r2	r3	r4	b1	b2	b3	b4
EM1	-1	0	0	0	1	-1	0	0
EM2	0	1	0	1	1	0	1	1
EM3	0	0	1	0	0	-1	0	1
EM4	-1	0	1	0	1	0	0	1

Figure 5.1: The elementary modes of an example network, where A, B, C, D and E (cycles) are given as metabolites which need fulfill a steady-state. Double-headed arrows labeled as r_1 and b_2 represent reversible reactions. Unfilled arrows labeled as r_2, r_3, r_4, b_1, b_3 and b_4 represent irreversible reactions. Elementary modes of this example are given in the table, where each row represents an elementary mode in which value 0 means that the corresponding reactions are not included in this elementary mode. (See also an metatool format of this example in supplementary materials which can be directly used for open software.)

reactions inactivated by an iterative procedure, mimicking a cascade of failures. [86] borrowed the concept of topological impact degree of [45] and extended it to deal with cycles in metabolic networks. However, these methods can not properly handle reversible reactions.

In this study, we propose a new model to assess the impact of gene perturbations on a metabolic network, together with efficient algorithms to compute it of large-scale networks. The model, which we call *flux balance impact degree* (FBID), builds upon the concept of steady-state fluxes and variability of FBA and FVA. The FBID of a perturbation is defined as the number of reactions which become inactive in all steady states after perturbation. We show that the FBID can be computed either by enumerating all EMs of the metabolic network, or by solving a series of linear programs, the later scaling much better to large networks. In contrast to techniques like FBA, FVA and MOMA, the new FBID does not require the definition of a specific objective function to model growth. Experiments on the *Escherichia coli* metabolic network show that FBID is competitive with existing approaches. It is computationally efficient even for global metabolic networks, where it outperforms existing approaches in terms of prediction accuracy. In addition, we provide supplementary information including detailed datasets, results and source code of this study[1].

[1]https://sunflower.kuicr.kyoto-u.ac.jp/ tyoyo/fbid/index.html

5.2 Flux balance impact degree (FBID)

We represent a metabolic network by its $m \times n$ stoichiometric matrix $\mathbf{S}$, where m is the number of metabolites and n is the number of reactions in the network. The activity of the network is represented by a flux vector $\mathbf{x} \in \mathbb{R}^n$ which contains all internal and exchange reactions in the network. A metabolic network for which mass balance constraints are satisfied is assumed to be in steady state, meaning that the flux vector satisfies:

$$\mathbf{S} \cdot \mathbf{x} = 0\,. \tag{5.1}$$

In addition, flux vectors must satisfy additional constraints of the form $\mathbf{a} \leq \mathbf{x} \leq \mathbf{b}$, where $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ are lower/upper limits for the fluxes in the network, to account to various constraints in the system. In particular, we can use them to encode the reversibility or irreversibility of reactions by setting the value of lower limits $\mathbf{a} \in \mathbb{R}^n$ to be -1 for reversible reactions and 0 for irreversible ones, while the upper limits $\mathbf{b}$ are set to 1. This ensures that fluxes are bounded by $[-1, 1]$ for reversible reactions, and by $[0, 1]$ for irreversible ones.

The metabolic networks we consider are usually under-determined since there are usually more reactions than metabolites ($n > m$). The set of admissible steady-state fluxes of the network is then the convex polytope:

$$\mathcal{A} = \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{S} \cdot \mathbf{x} = 0 \text{ and } \mathbf{a} \leq \mathbf{x} \leq \mathbf{b}\}\,. \tag{5.2}$$

Note that we assume that all reactions can be activated at steady state, meaning that for each reaction $i \in [1, n]$ there exists a flux vector $\mathbf{x}$ in $\mathcal{A}$ which satisfies $x_i \neq 0$. If this is not the case, we just remove the corresponding reactions from the network.

The perturbations we consider lead to gene knock-out, either by drug action or through DNA mutations. In our formalism, we represent a perturbation as a subset $\mathcal{R} \subset [1, n]$ of reactions inhibited by the perturbation. Inhibiting one or several reactions reduces their fluxes to zero in any steady state, and therefore reduces the set of admissible steady-state fluxes (5.2) to the reduced feasible set:

$$\mathcal{A}_{\mathcal{R}} = \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{S} \cdot \mathbf{x} = 0, \mathbf{a} \leq \mathbf{x} \leq \mathbf{b} \text{ and } x_i = 0, \forall i \in \mathcal{R}\}. \tag{5.3}$$

We can now formally define a new notion of flux balance impact degree of a perturbation, as the number of reactions which are inhibited in any steady-state following the perturbation:

Definition 5.2.1. *A reaction $j \in [1, n]$ is impacted by a perturbation $\mathcal{R} \subset [1, n]$ if $x_j = 0$ holds for all $\mathbf{x} \in \mathcal{A}_{\mathcal{R}}$. The* flux balance impact degree *(FBID) of $\mathcal{R}$ is the number of reactions impacted by $\mathcal{R}$.*

We note that, by definition, all reactions in $\mathcal{R}$ are impacted by $\mathcal{R}$, and therefore the FBID of $\mathcal{R}$ is at least as large as the cardinality of $\mathcal{R}$ itself. It can be strictly larger when other reactions, not directly in $\mathcal{R}$, are directly or indirectly affected by the knock-out of $\mathcal{R}$. For example, in the network represented on Figure 5.1, let $\mathcal{R} = \{r_4\}$. We see that reactions r_2 and b_3 affected by the knock-out of $\mathcal{R}$, and therefore the FBID of $\{r_4\}$ is 3, the total number of inhibited reactions.

5.3 Methods

5.3.1 Elementary mode (EM)-based computation

In this section, we show how to compute the FBID of any perturbation $\mathcal{R}$ from the enumeration of the EMs of the network. Following [73], we recall that an EM is a minimal set of reactions that allows a metabolic network to function in a steady state, i.e., a minimal set of reactions $e_i \subset [1, n]$ such that there exists a flux vector $\mathbf{e}_i \in \mathcal{A}$ satisfying the condition $\mathbf{e}_i(k) = 0$ if and only if $k \notin e_i$, where $\mathbf{e}_i(k)$ denotes the flux value of reaction k in the flux vector $\mathbf{e}_i$. Interestingly, the set E of all EMs of a metabolic network forms a basis of admissible steady state fluxes [75].

We now propose an algorithm to compute the FBID of a perturbation $\mathcal{R}$ from the list E of EMs of a metabolic network:

1. Compute the set of EMs E of the given metabolic network.

2. For a perturbation $\mathcal{R} \subset [1, n]$, select the subset of EMs from E which do not contain reactions in $\mathcal{R}$, that is:

$$E_{\mathcal{R}} = \{e_i \in E : e_i \cap \mathcal{R} = \emptyset\} . \tag{5.4}$$

3. The set of reactions $\mathcal{I}_{\mathcal{R}}$ impacted by $\mathcal{R}$ is computed as the set of reactions which are not contained in any EMs of $E_{\mathcal{R}}$:

$$\mathcal{I}_{\mathcal{R}} = [1, n] \setminus \bigcup_{e_i \in E_{\mathcal{R}}} e_i . \tag{5.5}$$

We now prove that this algorithm is correct, in the sense that the set $\mathcal{I}_{\mathcal{R}}$ it outputs in (5.5) is precisely the set of reactions impacted by $\mathcal{R}$ in the sense of Definition 5.2.1. Let us first consider a reaction $i \in [1, n]$ which is not in $\mathcal{I}_{\mathcal{R}}$. From (5.5) there exists an EM $e \in E_{\mathcal{R}}$ such that $i \in e$. The flux vector $\mathbf{e}$ corresponding to e is by definition admissible and has zero flux on the perturbed reactions by (5.4). It therefore belongs to $\mathcal{A}_{\mathcal{R}}$, and since it has a non-zero flux on reaction i, this reaction is not impacted by $\mathcal{R}$ according to Definition 5.2.1. This shows that all impacted

reactions are in $\mathcal{I}_\mathcal{R}$. Conversely, let us consider a reaction i which is not impacted by $\mathcal{R}$ in the sense of Definition 5.2.1. This means that there exists a flux $\mathbf{x} \in \mathcal{A}_\mathcal{R}$ such that $\mathbf{x}(i) \neq 0$. However, by Lemma 1 of [75], since $\mathbf{x}(j) = 0$ for $j \in \mathcal{R}$ it can be decomposed as a linear combinations of EMs which have themselves zero flux on $\mathcal{R}$, meaning that it can be decomposed as a linear combination of EMs in $E_\mathcal{R}$. Since $\mathbf{x}(i) \neq 0$ there must be at least an EM in $E_\mathcal{R}$ with non-zero flux in i, meaning that $i \notin \mathcal{I}_\mathcal{R}$. This shows that all reactions in $\mathcal{I}_\mathcal{R}$ are impacted, which concludes the proof.

To run this algorithm, we need to first compute all EMs of a network, which is a computational demanding task [28]. Although computation of all EMs of a given metabolic network may demand a high computation cost, this operation needs to be performed only once. The rest of computation (steps 2 and 3) can be done very fast. We use the example given in Fig. 5.1 to elucidate the step 2 and 3. Suppose perturbation $\mathcal{R} = \{r_2, b_1\}$ is given. Following the step 2, $E_\mathcal{R}$ is the subset with only one mode EM_2 (Figure 5.1: blue area of the table), because EM_1, EM_4 and EM_5 contain reaction b_1 and EM_1 and EM_3 contain both r_2. Then we only refer to $E_\mathcal{R}$ to compute the impacted reaction set as step 3. In this example, the impacted reactions are $\{r_1, r_2, r_4, b_1, b_3\}$ and the FBID of $\mathcal{R}$ is 5.

5.3.2 Linear programming (LP)-based computation

Since the reduced feasible set $\mathcal{A}_\mathcal{R}$ is defined in (5.3) by linear constraints, we propose an alternative algorithm to the EM-based approach based on LP to compute the FBID of a perturbation. Given a perturbation $\mathcal{R} \subset [1, n]$ and a reaction $i \in [1, n] \backslash \mathcal{R}$, we consider the following optimization problems to decide whether reaction i is impacted by $\mathcal{R}$:

$$\begin{array}{lll} \max & x_i & \\ \text{subject to} & \mathbf{S} \cdot \mathbf{x} = \mathbf{0} & \\ & x_j = 0, \forall j \in \mathcal{R} & \\ & \mathbf{a} \leq \mathbf{x} \leq \mathbf{b} & \end{array} \qquad \begin{array}{lll} \min & x_i & \\ \text{subject to} & \mathbf{S} \cdot \mathbf{x} = \mathbf{0} & \\ & x_j = 0, \forall j \in \mathcal{R} & \\ & \mathbf{a} \leq \mathbf{x} \leq \mathbf{b} & \end{array}$$

In other words, we perform FVA following each gene knock-out. However, contrary to classical use of FVA [78], we are just interested here in assessing whether the solutions to both optimization problems are 0 or not. Indeed, it is easy to see that reaction i is impacted by perturbation $\mathcal{R}$ according to Definition 5.2.1 if and only if the solutions of both problems are equal to 0, because this means that in the feasible set of both problems which is exactly $\mathcal{A}_\mathcal{R}$, x_i is constrained to be 0.

In practice, to compute the FBID of a perturbation $\mathcal{R}$ containing K reactions, one should solve a total of $2(n - K)$ LP, corresponding to two problems for each reaction $i \in [1, n] \backslash \mathcal{R}$. Since each LP can be solved in polynomial time, we obtain

a polynomial time algorithm to compute the impact of all perturbations containing a bounded number of reactions. In addition, since all LP are related to each other, significant speed-up can be obtained by using warm restart, as implemented in the fastFVA software [30]. Further speed-up is also possible by solving batches of LP in parallel on a distributed computing environment.

5.4 Data

5.4.1 The Escherichia coli metabolic networks

We use three versions of the *Escherichia coli* (*E. coli*) metabolic network, as summarized in Table 5.1. The central network is from the KEGG database [46, 48], and iJE660 and iJO1366 are from the BiGG database [62, 72], stored as METATOOL and SBML formats respectively. iJO1366 is the latest version of *E. coli* network, while we keep the older iJE660 in our experiments to allow comparison with previous work [21, 22, 68].

Table 5.1: The *E. coli* network with different versions.

Versions	# Reactions	# Metabolites	# Genes
Central network	63	59	85
iJE660	627	438	660
iJO1366	2251	1136	1366

We should notice that these networks are obtained as closed systems, and thus, additional information like sources and biomass synthetics is needed to make these systems open [21, 68, 92]. Sources provide compounds to be consumed while biomass synthetics are compounds exhausted by the networks. In the implementation, we use two types of sources, detailed in the supplementary materials. The first source represents a minimal medium consisting mainly of energy source, carbon dioxide and oxygen. The other source is a rich environment which covers the minimal medium together with 20 amino acids, biotin, thiamin and riboflavin, etc. The *E. coli* output biomass is given also in the supplementary materials.

5.4.2 Phenotypic data

In order to compare our *in silico* impact predictions to experimental data, we consider five datasets used in previous studies to assess the phenotypic consequences of gene knock-out.

The first dataset, collected from literature by [21], measures the growth capability of 79 gene deletion mutants, among which 41 are essential, 36 are non-essential, and 2 have been observed as either essential or non-essential. Following [92], we consider the predictions of any method on the later 2 genes as always correct in order to compute the accuracy of the prediction, while we remove them to compute ROC curves.

The second dataset (*insertional mutants*), collected by [7] and further used by [92], gives the growth rate of 481 mutants obtained by knock-out of single genes, among which 222 with more than 50% decrease in growth rate are considered essential. While all genes are available in the iJE660 network, only 461 (including 218 essentials) are present in the iJO1366 network.

The third data set is the combination of the first two ones. Although they contain genes in common, we follow [92] and consider them all different since they are part of different networks specific to each dataset.

The fourth dataset, collected by [29] and further used by [92], evaluates the gene variability of 598 mutants, among which 120 are considered essential. While all genes are available in the iJE660 network, only 571 (including 117 essentials) are present in the iJO1366 network.

The fifth dataset is the KEIO collection, collected by [6], which partitions 4288 mutants into 317 [including 14 newly added by 94] essential and 3971 nonessential genes. Among them, 81 (resp. 144) essential and 554 (resp. 1222) nonessential genes are present in the iJE660 (resp. iJO1366) model.

Because these experimental data sets are under different conditions, we used different input sources and output biomass in the networks for the different datasets, as listed in supplementary files. In short, for the mutants collected from literature, the minimal medium set is used to reconstruct 4 distinct networks, each of which includes only one of the energy sources. For the insertional mutants, we reconstruct the network by adding the minimal medium input with all energy sources. We use the rich source set when analyzing the Gerdes data set and KEIO collection. As for outputs, all analyses with iJE660 share the same biomass output (Table 3 in supplementary files), while for iJO1366, we use the core growth biomass proposed by [62].

5.5 Results

5.5.1 Implementation

We both used fastFVA [30] and ILOG CPLEX (version 11.2)[2] to solve the LP instances of the LP-based method, and CellNetAnalyzer which is a free software

[2]http://www.ilog.com/products/cplex

running under MATLAB to compute the EMs of a network [53]. All computations were performed on a PC with a Xeon CPU 3.33 GHz and 10 GB RAM running under the LINUX OS.

For each of the three *E. coli* metabolic networks listed in Table 5.1, we computed the FBID of each single gene deletion. Note that since a gene can catalyze several reactions, the perturbation set $\mathcal{R}$ associated to a gene deletion is the set of reactions catalyzed by the gene. We first assess the computational performance of the approach, before assessing the ability of FBID to predict phenotypes and compare it to state-of-the-art methods.

5.5.2 Computation time

We proposed two algorithms to compute the FBID of a perturbation: one approach based on enumerating EM (Section 5.3.1), and one approach based on an LP formulation (Section 5.3.2). Table 5.2 shows the total computation time to perform the experiment on each network. For the LP-based approach, this is the total time to solve all LP with fastFVA; for the EM-based method, this is the time to compute the EMs of each network once with CellNetAnalyzer, and then output the list of impacted reactions for each gene deletion. We see that the EM-based method is

Table 5.2: Computational time for FBID computation.

	Computational time (s)	
Versions	LP-based	EM-based
Central network	8	4
iJE660	252	$>$ 7 days
iJO1366	10234	$>$ 7 days

very fast for a small network but not efficient for large ones; in fact CellNetAnalyzer did not manage to compute the EMs of both large networks within a week. This is coherent with the exponential complexity of the method. On the other hand, although many LP instances need to be solved for the LP-based method, we see that its polynomial complexity allows it to better scale to large networks. fastFVA [30] manages to finish all computations on the largest network with 2251 reactions within around 3 hours, and is roughly two orders of magnitude faster than a naive implementation solving all LP instances independently from each other with CPLEX (see supplementary information).

Based on these observations, in what follows we only run the LP-based implementation with fastFVA to compute the FBIDs corresponding to the different genes and networks investigated for phenotypic prediction. The total computation times

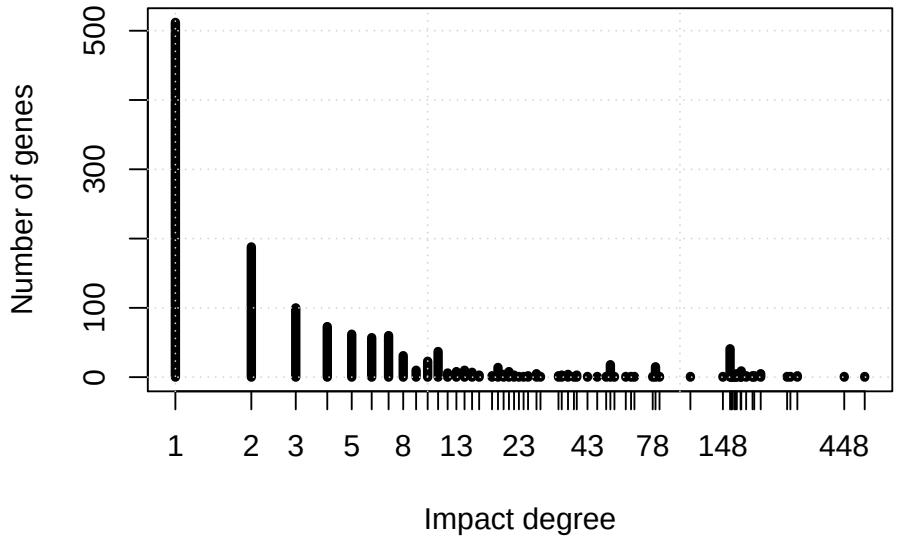


Figure 5.2: FBID distribution for the 1366 genes of the KEIO collection dataset computed on the iJO1366 metabolic network.

for both *E. coli* global metabolic networks (iJE660 and iJO1366) are summarized in the supplementary materials.

5.5.3 Phenotypic prediction

The FBIDs computed on each network vary significantly between different genes. For example, Figure 5.2 shows the distribution of FBIDs for the 1366 genes of the KEIO collection dataset estimated on the iJO1366 model. While more than 80% of all genes have an FBID smaller than 10, it increases to 540 for the *msbA* (b0914) gene, a bacterial lipid flippase whose knock-out blocks ATP synthesis by oxidative phosphorylation, or 448 for *acpP* (b1094), a acyl carrier protein which catalyzes polyketide biosynthesis of holo-ACPS; unsurprisingly, both are essential genes.

To assess more quantitatively how predictive the FBID is for gene essentiality, we systematically compared the FBID corresponding to each gene deletion to the corresponding experimental phenotypic data, for both versions of the large *E. coli* metabolic network (iJE660 and iJO1366). In each experimental data, the genes are separated in two classes corresponding to genes with a large or small phenotypic impact. By thresholding the FBID to some level, we can predict that genes with an FBID above the threshold should have large phenotypic impact, while those below the threshold should not. Figure 5.3 shows the receiver operating characteristic (ROC) curve for each dataset and each network, corresponding to the sensitivity

plotted as a function of 1-specificity when we vary the FBID threshold. In addition, we show on Table 5.3 the area under the ROC curve (AUC) and the accuracy reached in each case, when the FBID threshold is set for each phenotypic dataset to maximize the accuracy as in [92].

We can see that phenotype prediction for the iJE660 and iJO1366 *E.coli* networks are overall similar, with an advantage for the former on all phenotype datasets. The performance on the first dataset (collected from literature) is rather disappointing. This can be explained, to some extent, by the fact that we had to modify the network by using the minimal inputs together with distinct carbon sources, which resulted in many metabolites and reactions being always inactive at steady state. The performance on the insertional mutants dataset is also not very good and may also be due in part to the particular context of using minimal inputs. For the Gerdes dataset and KEIO collection, FBID performs pretty well on both networks reaching an AUC of 0.66 and 0.78 for iJE660 and 0.6 and 0.72 for iJO1366, respectively.

Table 5.3: Performance of FBID on gene essentiality prediction, using both iJE660 and iJO1366.

	AUC		Accuracy	
Experimental data	iJE660	iJO1366	iJE660	iJO1366
Collected from literature	0.57	0.49	68%	63%
Insertional mutants	0.55	0.50	57%	52%
Combined data set	0.57	0.49	59%	54%
Gerdes data set	0.66	0.60	82%	83%
KEIO collection	0.78	0.72	89%	93%

In order to compare the performance of FBID with existing approaches, we first focus on the iJE660 *E. coli* network which was used by [92] to compare SA [92], FBA [22], MOMA [76] and EMA [84]. Results are summarized in Table 5.4, where we directly report the accuracies provided by [92] for existing methods.

On the mutants collected from literature, our approach based on FBID is clearly worse than SA, FBA and EMA, which reach very high accuracy (90% for EMA). This can be explained, to some extent, because this collection includes genes which only catalyze the central metabolism [21] where alternative paths are numerous when we block a single gene. Therefore, although changes in optimal fluxes captured by FBA, or decrease in number of EMs captured by EMA correlate well with growth rate, our approach meets difficulties in finding important fluctuations in the number of reactions that become completely inhibited when a gene is deleted. On the insertional mutant dataset, all methods reach a similar level of accuracy, with a

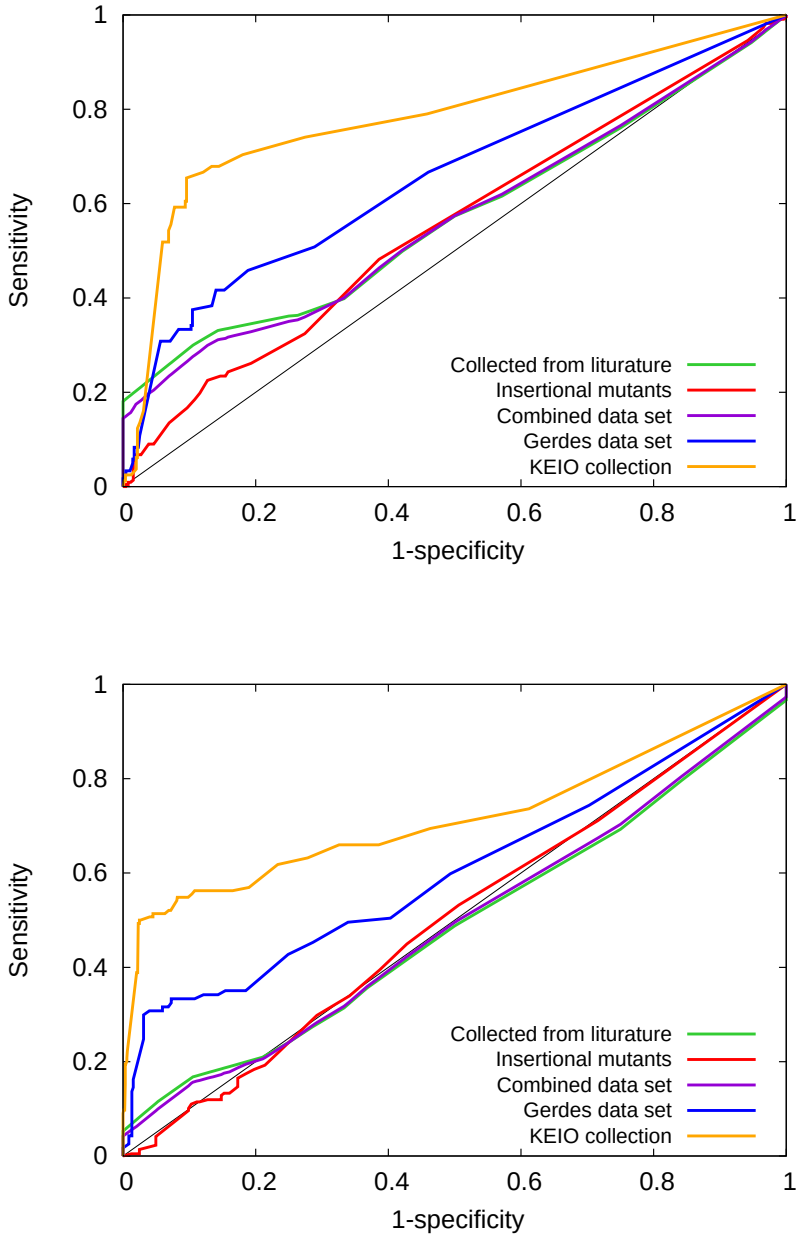


Figure 5.3: ROC curves for phenotype prediction from the FBID on various datasets, using both the iJE660 metabolic network (top) and the larger iJO1366 network (bottom).

slight advantage for SA and MOMA over FBA and FBID. On the larger Gerdes dataset, FBID and FBA reach the same level, and clearly outperform SA.

Table 5.4: Comparison of the accuracy of FBID with different methods using the iJE660 network.

	Method				
Experimental data	FBID	SA	FBA	MOMA	EMA
Collected from literature	68%	71%	86%	-	90%
Insertional mutants	57%	60%	58%	59%	-
Combined data set	59%	62%	62%	-	-
Gerdes data set	82%	74%	82%	-	-

In order to further investigate the performance of FBID on large networks, we compare it to FBA on the largest iJO1366 network for the prediction of gene essentiality as defined in the KEIO collection. Figure 5.4 shows the ROC and precision-recall curves of both methods. We see that FBID (AUC=0.72, accuracy=93%) outperforms FBA (AUC=0.68, accuracy=89%) on this experiment, confirming the potential of FBID on large networks.

As shown on Figure 5.5, the predictions of FBID and FBA are correlated: genes with a large FBID (on the right) often have a small FBA score (near the bottom), corresponding to two notions of essentiality. However, the correlation is not perfect, and we observe for example a number of non-essential genes with a small FBA scores and a small FBID (near the bottom left); in that case, the FBID is a better indicator of essentiality. Another advantage of FBID over FBA is the fact that FBA has difficulties to make a difference between the genes predicted to be essential. For example, 109 genes out of 1322 have a minimum FBA score of 0, corresponding to a complete blockage of fluxes; however, only 48 of them (44%) are truly essential. This means that FBA can not predict essentiality with more than 44% precision, as can be seen on the precision-recall curve (Figure 5.4). On the contrary, FBID is better able to rank the genes with large scores, and can reach much higher precision than FBA near the top of the list. This is particularly relevant for applications where we want to predict a few essential genes with high precision. More details about FBA and FBID essentiality prediction can be found in the supplementary information.

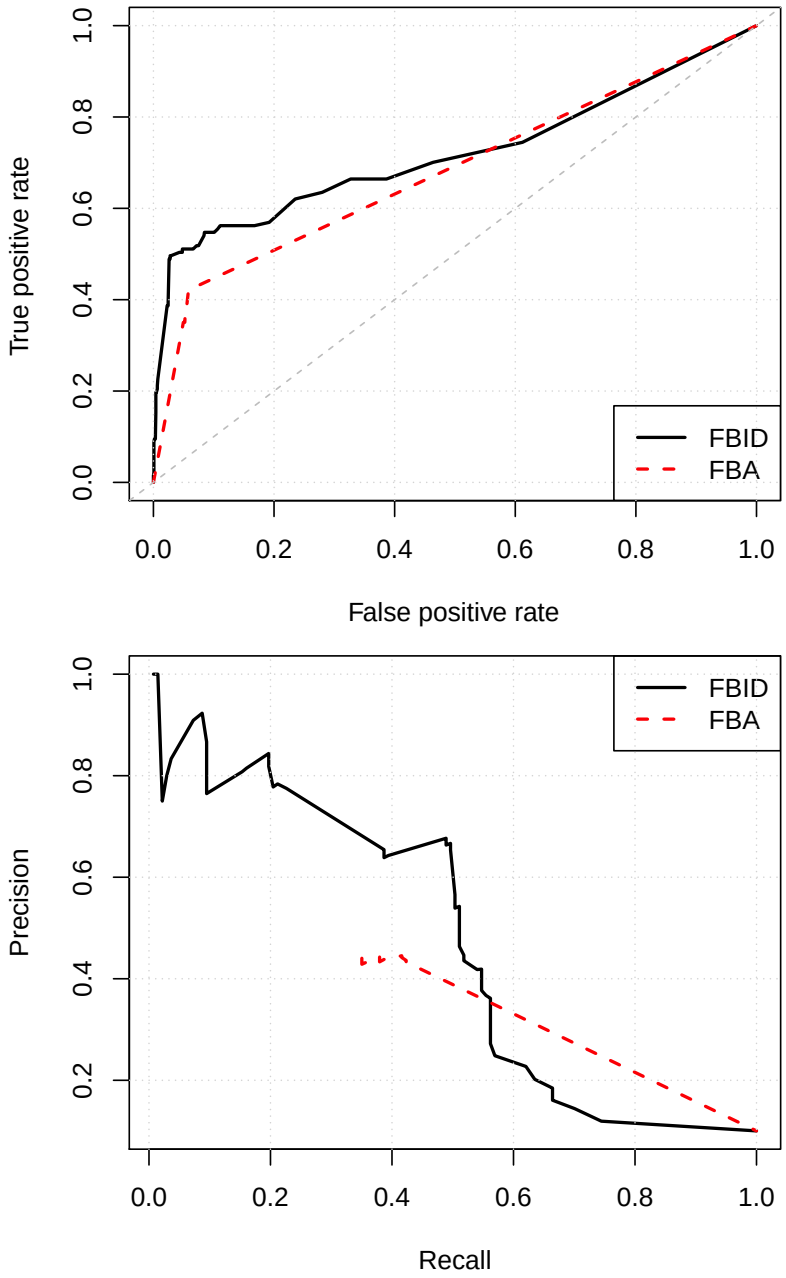


Figure 5.4: ROC curve (top) and precision-recall curve (bottom) for phenotype prediction with FBID and FBA on the Keio dataset using the iJO1366 network .

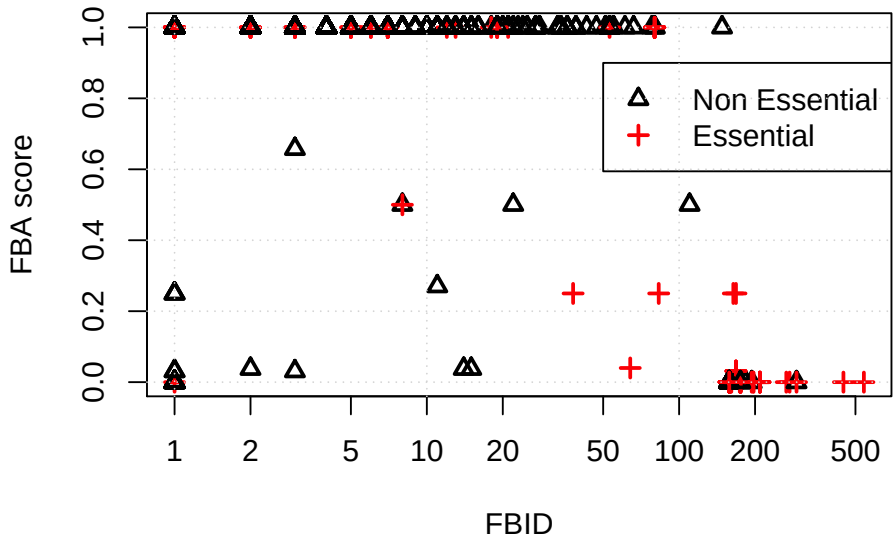


Figure 5.5: FBID and FBA scores for all genes in the Keio dataset analyzed with the iJO1366 network. Red circles correspond to experimentally essential genes.

5.6 Discussion

We have proposed FBID, a new definition of impact degree which can not only efficiently deal with the reversible reactions in metabolic networks but also have the state conditions being taken into account. To compute the FBID against perturbations, we proposed two algorithms, an LP-based method and an EM-based algorithm. The advantage of the LP-based method is that it can solve all LP instances individually, can strongly benefit from warm restart techniques and is amenable to parallelization. Contrary to other LP-based formalisms like FBA, FVA or MOMA, it does not depend on a subjective definition of a relevant objective function. Although computational cost of the LP-based method grows with the network size and the number of perturbations to be tested, the overall time complexity is still bounded polynomially. If we are interested in only a few candidate perturbations, then only the corresponding LP need to be solved. The EM-based method, on the other hand, can compute the FBID of specific perturbations very fast for middle-scale networks. The main computational advantage of this approach is that the computation of EMs needs to be performed only once, no matter how many perturbation we want to test – including perturbations involving several reactions. This advantage vanishes for large-scale metabolic networks, however, because of the exponential complexity of computing EMs and the lack of efficient algorithms for that purpose.

We carried out computational experiments by using *E. coli* metabolic networks. The results on computational time for calculating the FBID of different sized networks show that the LP-based method implemented with fastFVA is very efficient while the EM-based method did not return any result for large networks due to the difficulty of computing the EMs. In terms of phenotype prediction, we obtained poor results when we tested metabolic networks with a minimal source input, because many metabolic paths are always closed in this case. Comparison of the performance of phenotype prediction with some existing methods indicates that the FBID performs as well as other models or even better, particularly on large networks.

The interpretation we give of the FBID in terms of EMs makes an interesting link with existing EM-based methods that measure how many EMs disappear when we inhibit a reaction [90, 12]. In our case, we also enumerate the list of EMs that remain once the reaction is inhibited, but instead of focusing on the *number of EMs* remaining, we focus instead on the *number of reactions* that can still be activated in the remaining EMs. Although the number of EMs in a network has been used as a measure of flexibility and as an estimate of fault-tolerance [84], we propose here that the amount of reactions inactivated in cascade may be a better indicator of gene essentiality. Of course, the *number* of reactions inactivated is itself a very crude measure, and investigating variants such as weighting reactions by their 'importance' before counting them may be interesting future work.

Chapter 6

Conclusion and future work

6.1 Summary

In this thesis, we presented novel methods for analysis of chemical graphs and biological network structures, which deal with four types of problems. Since the simplicity of tree-structured data, our first two approaches (in Chapter 2 and Chapter 3) proposed methods for grammar-based tree compression and tree-like molecular enumeration, which was successfully applied to pattern extraction of glycans and provided a vast potentiality to improve our today's drugs by new compound design, respectively. Next, we focused on more complex biological systems, and both presented efficient methods for prediction of protein complexes (in Chapter 4) and developed a new model for analyzing the robustness of metabolic networks (in Chapter 5).

In Chapter 2, we have proposed integer programming-based methods for finding the minimum grammars of strings and (un)ordered trees. The results of artificial experiments indicated that our proposed methods are correct and efficient for ordered tree compression, while some improvements are required for unordered trees. Moreover, we obtained twelve glycans from KEGG databased to perform some real-world experiments. The results showed that our IP-based grammar tree compression can efficiently extract interesting patterns. Comparison of the generated grammar size with an existing method indicated that our proposed methods performs better with the compression ratio than state-of-the-art one.

In Chapter 3, we have developed two methods for enumerating tree-like compounds with and without multiple bonds by firstly using breadth first search order. To reduce the exponentially increasing search space with increasing atoms, we employed some important concepts such as center-rooted, left-heavy and normal form. Owing to these concepts, our methods can only produce balanced intermediate trees during their search of a family tree without proceeding unbalanced ones so as to

avoid duplicates. We also performed some computational experiments, and the results showed that the proposed methods are exact and faster than state-of-the-art ones. Furthermore, we proceeded structural matching with the generated structures by finding them in biological database. The low matching rates indicated that a large amount of structures have been exponentially generated, but few of them have been discovered to represent real materials. This finding is encouraging since understanding of such unexplored chemical structures can help to discover new compounds with desired property. And indeed, exploration of these unmatched chemical structures also plays an important role in today's drug design.

In Chapter 4, we have presented an improved integer programming based-method and its combination methods with maximal components and extreme sets for verifying candidate protein complexes predicted by graph clustering methods from large-scale protein-protein interaction networks. The basic idea is that a candidate protein complex should not be divided into many small complexes. We also prove that our problem of maximizing the size of a connected component given by verified protein-protein interactions is NP-hard. Then, we implemented this improved IP-based method and the combination methods with maximal components and extreme sets to perform several computational experiments. Comparison with the existing method is also conducted to confirm the advantage of our methods. Finally, we discuss the results of our proposed methods.

In Chapter 5, we have proposed FBID, a new definition of impact degree, which can not only efficiently deal with the reversible reactions but also have state conditions being taken into account. This model assesses the impact of gene perturbations on a metabolic network, and its definition against a perturbation is defined as the number of inhibited reactions in all steady state after the perturbation. To compute FBID against perturbations, we proposed two algorithms, an LP-based method and an EM-based method. Different from other formalisms, our LP-based method does not depend on a subjective definition of a relevant objective function. Other advantages of this method goes that all LP instances can be solved individually and the overall time complexity is bounded polynomially. On the other hand, the EM-based method can compute the FBID of specific perturbations fast for middle-scale networks, since the computation of EMs needs to be performed only once. However, this advantage only vanishes for large-scale networks because of the exponential time complexity of computing EMs. In addition, we performed computational experiments on some *E. coli* metabolic networks. The computational time for calculating the FBID of distinct sized networks showed that the LP-based method solved by fastFVA is efficient for large-scale networks, while the EM-method did not return any results due to its difficulty for computation of EMs. Comparison of the results of phenotype prediction with some existing methods indicated that the FBID performs as well as other models or even better, especially for large networks.

6.2 Future directions

In this thesis, we have focused on analyzing structural topology of biological networks for the purpose of further understanding of their physiological function and organic mechanisms. We have developed several useful methodologies towards four type of problems in system biology. Our study started with the analysis of the tree-structured biological data and then spread to deal with complex biological (cyclic) networks via computational algorithms and network models. Although the results of series of computational experiments have indicated the outperformance of our methods against state-of-the-art ones, further improvements for these approaches are still required.

In the first part of this thesis, although our proposed methods can efficiently deal with tree-structured data compression and tree-like molecular enumeration, they cannot handle cyclic structures. Since there are enormous amount of structures in bioinformatics databases that have cycles, it is important to find new structural properties to extend our methods for dealing with more complex structures. Particularly in Chapter 3, our proposed methods are not limited to tree-like molecular enumeration, and it is feasible that our enumeration algorithms can be extended to deal with cyclic compounds by just representing these cyclic sub-structures as special vertices. For example, benzene can be represented as an atom with valence 6, together with some restrictions for distinguishing its 6 positions which aims to avoid duplicates. Moreover, further combination of chemical activity and biological properties with our enumeration methods should be an interesting future step for further prediction of new compounds or reduction of unreal isomers from the large unexplored chemical space.

In the second part of this thesis, although we have shown the hardness of the problem of verifying candidate protein complexes, and have proposed an improved IP-based method together with its combination methods for solving it, it still needs some improvements (Chapter 4, Section 4.5). Since we have solved this verification problem approximately, we need to investigate the compact formulation of the problem of maximizing the size of a connected component from candidate complexes. Other future work is required to improve the recall and the efficiency factor of our methods to a higher range.

In Chapter 5, the interpretation we gave of the FBID has shown to be a better indicator of gene essentiality than state-of-the-art models when we carried out the application of phenotype prediction. However, the impact against a perturbation cannot be determined only by the *number* of inactivated reactions which is itself a crude measure, since each reaction in a metabolic network has slightly different biological function. Therefore, investigating variants such as weighting reactions by their '*importance*' before counting them may be interesting future work. In terms of such variants, we aim to extend our model to '*weighted*-FBID', an improved new

definition of impact degree against perturbations, which is expected to be a better descriptor of the robustness of metabolic networks with respect to gene knock-out. The *weighted*-FBID of a perturbation is supposed to be defined as a summation of the '*importance*' values of reactions which become inactive in all steady states after perturbation. There are several estimators in graph theory (such as degree centrality, betweenness and eigenvector, etc.) that can measure the relative importance of vertices within a network. However, further investigation of these measures is needed to well assess the suitable '*importance*' values for reactions in a metabolic network, such that combinations of such measures with our FBID would get a better performance for the analysis of the robustness of metabolic networks.

Bibliography

[1] Acuna, V., Chierichetti, F., Lacroix, V., Spaccamela, A. M., Sagot, M. F., and Stougie, L. Modes and cuts in metabolic networks: Complexity and algorithm. *Biosystems*, 95(1):51–60, 2009.

[2] Akutsu, T. A bisection algorithm for grammar-based compression of ordered trees. *Inform. Process. Lett.*, 110:815–820, 2010.

[3] Akutsu, T., Fukagawa, D., Jansson, J., and Sadakane, K. Inferring a graph from path frequency. *Discrete Appl. Math.*, 160:1416–1428, 2012.

[4] Akutsu, T. and Nagamochi, H. Comparison and enumeration of chemical graphs. *Computational and Structural Biotechnology Journal*, 5(6):e201302004, 2013.

[5] Albert, R. Scale-free networks in cell biology. *J. Cell Sci.*, 118:4947–4957, 2005.

[6] Baba, T., Ara, T., Hasegawa, M., Takai, Y., Okumura, Y., Baba, M., Datsenko, K. A., Tomita, M., Wanner, B. L., and Mori, H. Construction of escherichia coli k-12 in-frame, single-gene knockout mutants: the keio collection. *Mol. Syst. Biol.*, 2(2006.0008):1–11, 2006.

[7] Badarinarayana, V., Estep, P. W., Shendure, J., Edwards, J., Tavazoie, S., Lam, F., and Church, G. M. Selection analyses of insertional mutants using subgenic-resolution arrays. *Nat. Biotechnol.*, 19:1060–1065, 2001.

[8] Bader, G. D. and Hogue, C. W. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4:2, 2003.

[9] Ballerstein, K., Kamp, A. V., Klamt, S., and Haus, U. U. Minimal cut sets in a metabolic network are elementary modes in a dual network. *Bioinformatics*, 28(3):381–387, 2012.

[10] Barkai, N. and Leibler, S. Robustness in simple biochemical networks. *Nature*, 387:913–917, 1997.

[11] Bateman, A., Coin, L., Durbin, R., Finn, R. D., Holich, V., Griffiths-Jones, S., Khanna, A., Marshall, M., Moxon, S., Sonnhammer, E. L. L., Studholme, D. J., Yeats, C., and Eddy, S. R. The Pfam protein families database. *Nucleic Acids Res.*, 32:D138–D141, 2004.

[12] Behre, J., Wilhelm, T., von Kamp, A., Ruppin, E., and Schuster, S. Structural robustness of metabolic networks with respect to multiple knockouts. *J. Theor. Biol.*, 252(3):433–441, 2008.

[13] Brohée, S. and van Helden, J. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, 7:488, 2006.

[14] Busatto, G., Lohrey, M., and Maneth, S. Efficient memory representation of xml document trees. *Inform. Syst.*, 33:456–474, 2008.

[15] Charikar, M., Lehman, E., Liu, D., Panigrahy, R., Prabhakaran, M., Sahai, A., and Shelat, A. The smallest grammar problem. *IEEE Trans. Inform. Theor.*, 51:2554–2576, 2005.

[16] Chen, J., Hsu, W., Lee, M. L., and Ng, S. K. Discovering reliable protein interactions from high-throughput experimental data using network topology. *Artif. Intell. Med.*, pages 37–47, 2005.

[17] Chua, H. N., Ning, K., Sung, W. K., Leong, H. W., and Wong, L. Using indirect protein-protein interactions for protein complex prediction. *J. Bioinformatics Computat. Biol.*, 6(3):435–466, 2008.

[18] Consortium, U. Ongoing and future developments at the universal protein resource. *Nucleic Acids Res.*, 39:D214–D219, 2011.

[19] Deshpande, M., Kuramochi, M., Wale, N., and Karypis, G. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Trans. Knowl. Data Eng.*, 17:1036–1050, 2005.

[20] Dobson, C. M. Chemical space and biology. *Nature*, 432:824–828, 2004.

[21] Edwards, J. S. and Palsson, B. O. The *Escherichia coli* mg1655 in silico metabolic genotype: its definition, characteristics, and capabilities. *Proc. Natl. Acad. Sci. Unit States Am.*, 97(10):5528–5533, 2000.

[22] Edwards, J. S. and Palsson, B. O. Robustness analysis of the escherichia coli metabolic network. *Biotechnol. Prog.*, 16(6):927–939, 2000.

[23] Enright, A. J., Dongen, S. V., and Ouzounis, C. A. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.*, 30(7):1575–1584, 2002.

[24] Faulon, J. L. and Bender, A. *Handbook of Chemoinformatics Algorithms*. CRC Press, 2010.

[25] Faulon, J. L., Jr., D. P. V., and Roe, D. Enumerating molecules. *Reviews in Computational Chemistry*, 21:209–286, 2005.

[26] Finn, R. D., Marshall, M., and Bateman, A. iPfam: visualization of protein-protein interactions in PDB at domain and amino acid resolutions. *Bioinformatics*, 21(3):410–412, 2005.

[27] Fujiwara, H., Wang, J., Zhao, L., Nagamochi, H., and Akutsu, T. Enumerating treelike chemical graphs with given path frequency. *J. Chem. Inform. Model.*, 48(7):1345–1357, 2008.

[28] Gagneur, J. and Klamt, S. Computation of elementary modes: a unifying framework and the new binary approach. *BMC Bioinformatics*, 5(175):1–21, 2004.

[29] Gerdes, S. Y., Schille, M. D., Campbell, J. W., Balázsi, G., Ravasz, E., Daugherty, M. D., Somera, A. L., Kyrpides, N. C., Anderson, I., Gelfand, M. S., Bhattacharya, A., Kapatral, V., D'Souza, M., Baev, M. V., Grechkin, Y., Mseeh, F., Fonstein, M. Y., Overbeek, R., Barabási, A. L., Oltvai, Z. N., and Osterman, A. L. Experiment determination and system level analysis of essential genes in escherichia coli mg1655. *J. Bacteriol.*, 185(19):5673–5684, 2003.

[30] Gudmundsson, S. and Thiele, I. Computationally efficient flux variability analysis. *BMC Bioinformatics*, 11(489):1–3, 2010.

[31] Gugisch, R., Kerber, A., Kohnert, A., Laue, R., Meringer, M., Rucker, C., and Wassermann, A. *Molgen 5.0, a Molecular Structure Generator*. Bentham Science Publishers Ltd., 2012.

[32] Habibi, M., Eslahchi, C., and Wong, L. Protein complex prediction based on k-connected subgraphs in protein interaction network. *BMC Syst. Biol.*, 4:129, 2010.

[33] Handorf, T., Christian, N., Ebenhöh, O., and Kahn, D. An environmental perspective on metabolism. *J. Theor. Biol.*, 252(3):530–537, 2008.

[34] Hartwell, L. H., Hopfield, J. J., Leibler, S., and Murray, A. W. From molecular to modular cell biology. *Nature*, 402:C47–C52, 1999.

[35] Hashimoto, K., Goto, S., Kawano, S., Aoki-Kinoshita, K., Ueda, N., Hamajima, M., Kawasaki, T., and Kanehisa, M. Kegg as a glycome informatics resource. *Glycobiology*, 16(5):63R–70R, 2006.

[36] Haus, U. U., Klamt, S., and Stephen, T. Computing knock-out strategies in metabolic networks. *J. Comput. Biol.*, 15(3):259–268, 2008.

[37] Hayashida, M. and Akutsu, T. Image compression-based approach to measuring the similarity of protein structures. In *Proc. 6th Asia-Pacific Bioinformatics Conference*, pages 221–230, 2008.

[38] Hayashida, M. and Akutsu, T. Comparing biological networks via graph compression. *BMC Syst. Biol.*, 4(Suppl 2):S13, 2010.

[39] Hizukuri, Y., Yamanishi, Y., Nakamura, O., Yagi, F., Goto, S., and Kanehisa, M. Extraction of leukemia specific glycan motifs in humans by computational glycomics. *Carbohydr. Res.*, 340:2270–2278, 2005.

[40] Horváth, T. and Ramon, J. Efficient frequent connected subgraph mining in graphs of bounded tree-width. *Theor. Comput. Sci.*, 411:2784–2797, 2010.

[41] Imada, T., Ota, S., Nagamochi, H., and Akutsu, T. Efficient enumeration of stereoisomers of outerplanar chemical graphs using dynamic programming. *J. Chem. Inform. Model.*, 51:2788–2807, 2011.

[42] Ishida, Y., Kato, Y., Zhao, L., Nagamochi, H., and Akutsu, T. Branch-and-bound algorithms for enumerating treelike chemical graphs with given path frequency using detachment-cut. *J. Chem. Inform. Model.*, 50(5):934–946, 2010.

[43] Jeong, H., Mason, S. P., Barabási, A.-L., and Oltvai, Z. N. Lethality and centrality in protein networks. *Nature*, 411:41–42, 2001.

[44] Jiang, C., Coenen, F., and Zito, M. A survey of frequent subgraph mining algorithms. *Knowl. Eng. Rev.*, 28:75–105, 2013.

[45] Jiang, D., Zhou, S., and Chen, T. P. P. Compensatory ability to null mutation in metabolic networks. *Biotechnol. Bioeng.*, 102(2):361–369, 2009.

[46] Kanehisa, M. and Goto, S. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, 28:27–30, 2000.

[47] Kanehisa, M., Goto, S., Hattori, M., Aoki-Kinoshita, K. F., Itoh, M., Kawashima, S., Katayama, T., Araki, M., and Hirakawa, M. From genomics to chemical genomics: new developments in kegg. *Nucleic Acids Res.*, 34:D354–357, 2006.

[48] Kanehisa, M., Goto, S., Sato, Y., Furumichi, M., and Tanabe, M. Kegg for for integration and interpretation of large-scale molecular datasets. *Nucleic Acids Res.*, 40:D109–D114, 2012.

[49] Kiemer, L., Costa, S., Ueffing, M., and Cesareni, G. WI-PHI: A weighted yeast interactome enriched for direct physical interactions. *Proteomics*, 7:932–943, 2007.

[50] King, A. D., Pržulj, N., and Jurisica, I. Protein complex prediction via cost-based clustering. *Bioinformatics*, 20(17):3013–3020, 2004.

[51] Kitano, H. Computational systems biology. *Nature*, 420:206–210, 2002.

[52] Klamt, S. and Gilles, E. D. Minimal cut sets in biochemical reaction networks. *Bioinformatics*, 20(2):226–234, 2004.

[53] Klamt, S., Saez-Rodriguez, J., and Gilles, E. D. Structural and functional analysis of cellular networks with cellnetanalyzer. *BMC Syst. Biol.*, 1:2, 2007.

[54] Klamt, S. and Stelling, J. Combinatorial complexity of pathway analysis in metabolic networks. *Mol. Biol. Rep.*, 29(1-2):233–236, 2002.

[55] Lemke, N., Heredia, F., Barcellos, C. K., dos Reis, A. N., and Mombach, J. C. M. Essentiality and damage in metabolic networks. *Bioinformatics*, 20(1):115–119, 2004.

[56] Li, M., Badger, J., Chen, X., Kwong, S., Kearney, P., and Zhang, H. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17:149–154, 2001.

[57] Maruyama, O. and Chihara, A. NWE: Node-weighted expansion for protein complex prediction using random walk distances. *Proteome Science*, 9:S14, 2011.

[58] Maslov, S. and Sneppen, K. Specificity and stability in topology of protein networks. *Science*, 296(5569):910–913, 2002.

[59] Murakami, S., Doi, K., and Yamamoto, A. Finding frequent patterns from compressed tree-structure data. In *Proc. 11th Int. Conf. Discovery Science*, pages 284–295, 2008.

[60] Nagamochi, H. Graph algorithms for network connectivity problems. *J. Oper. Res. Soc. Jpn.*, 47:199–223, 2004.

[61] Nakano, S. and Uno, T. Generating colored trees. *Lect. Notes Comput. Sci.*, 3787:249–260, 2005.

[62] Orth, J. D., Conrad, T. M., Na, J., Lerman, J. A., H. Nam, A. M. F., and Palsson, B. A comprehensive genome-scale reconstruction of escherichia coli metabolism-2011. *Mol. Syst. Biol.*, 7(535):1–9, 2011.

[63] Ozawa, Y., Saito, R., Fujimori, S., Kashima, H., Ishizaka, M., Yanagawa, H., Miyamoto-Sato, E., and Tomita, M. Protein complex prediction via verifying and reconstructing the topology of domain-domain interactions. *BMC Bioinformatics*, 11:350, 2010.

[64] Pretsch, E., Bühlmann, P., and Badertscher, M. *Structure Determination of Organic Compounds*. Springer-Verlag Berlin Heidelberg, 2009.

[65] Pu, S., Wong, J., Turner, B., Cho, E., and Wodak, S. J. Up-to-date catalogues of yeast protein complexes. *Nucleic Acids Res.*, 37:825–831, 2009.

[66] Qi, Y., Balem, F., Faloutsos, C., Klein-Seetharaman, J., and Bar-Joseph, Z. Protein complexes identification by supervised graph local clustering. *Bioinformatics*, 24:i250–i258, 2008.

[67] Raman, K. and Chandra, N. Flux balance analysis of biological system: applications and challenges. *Brief. Bioinform.*, 10(4):435–449, 2009.

[68] Reed, J. L., Vo, T. D., Schilling, C. H., and Palsson, B. O. An expanded genome-scale model of escherichia coli k-12 (ijr904 gsm/gpr). *Genome Biol.*, 4(9):R54, 2003.

[69] Rouvray, D. H. The pioneering contributions of cayley and sylvester to the mathematical description of chemical structure. *J. Mol. Struct.*, 1:54, 1989.

[70] Rytter, W. Application of lempel-ziv factorization to the approximation of grammar-based compression. *Theor. Comput. Sci.*, 302:211–222, 2003.

[71] Sakamoto, H., Maruyama, S., Kida, T., and Shimozono, S. A space-saving approximation algorithm for grammar-based compression. *IEICE Trans. Info. Syst.*, 92-D:158–165, 2009.

[72] Schellenberger, J., Park, J. O., Conrad, T. C., and Palsson, B. Bigg: a biochemical genetic and genomic knowledgebase of large scale metabolic reconstructions. *BMC Bioinformatics*, 11:213, 2010.

[73] Schuster, S., Fell, D. A., and Dandekar, T. A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks. *Nat. Biotechnol.*, 18(3):326–332, 2000.

[74] Schuster, S. and Hilgetag, C. On elementary flux modes in biochemical reaction systems at steady state. *J. Biol. Syst.*, 2(2):165–182, 1994.

[75] Schuster, S., Hilgetag, C., Woods, J. H., and Fell, D. A. Reaction routes in biochemical reaction systems: Algebraic properties, validated calculation procedure and example from nucleotide metabolism. *J. Math. Biol.*, 45:153–181, 2002.

[76] Segre, D., Vitkup, E., and Church, G. M. Analysis of optimality in natural and perturbed metabolic networks. *Proc. Natl. Acad. Sci. Unit States Am.*, 99:15112–15117, 2002.

[77] Shimizu, M., Nagamochi, H., and Akutsu, T. Enumerating tree-like chemical graphs with given upper and lower bounds on path frequencies. *BMC Bioinformatics*, 12(Suppl 14):1–9, 2011.

[78] Shlomi, T., Cabili, M. N., and Ruppin, E. Predicting metabolic biomarkers of human inborn errors of metabolism. *Mol. Syst. Biol.*, 5:263, 2009.

[79] Smart, A. G., Amaral, L. A. N., and Ottino, J. M. Cascading failure and robustness in metabolic networks. *Proc. Natl. Acad. Sci. Unit States Am.*, 105(36):13223–13228, 2008.

[80] Spirin, V. and Mirny, L. A. Protein complexes and functional modules in molecular networks. *Proc. Natl. Acad. Sci. Unit States Am.*, 100:12123–12128, 2003.

[81] Spirin, V. and Mirny, L. A. Protein complexes and functional modules in molecular netwroks. *Proc. Natl. Acad. Sci. Unit States Am.*, 100(21):12123–12128, 2003.

[82] Sridhar, P., Song, B., Kahveci, T., and Ranka, S. Mining metabolic networks for optimal drug targets. *Pac. Symp. Biocomput.*, 13:291–302, 2008.

[83] Stark, C., Breitkreutz, B., Reguly, T., Boucher, L., Breitkreutz, A., and Tyers, M. BioGRID: a general repository for interaction datasets. *Nucleic Acids Res.*, 34:D535–D539, 2006.

[84] Stelling, J., Klamt, S., Bettenbrock, K., Schuster, S., and Gilles, E. D. Metabolic network structure determines key aspects of functionality and regulation. *Nature*, 420:190–193, 2002.

[85] Strogat, S. H. Exploring ecomplex networks. *Nature*, 410:268–276, 2001.

[86] Tamura, T., Cong, Y., Akutsu, T., and Ching, W. K. An efficient method of computing impact degrees for multiple reactions in metabolic networks with cycles. *IEICE Trans. Info. Syst.*, E94-D(12):2393–2399, 2011.

[87] Tamura, T., Takemoto, K., and Akutsu, T. Finding minimum reaction cuts of metabolic networks under a boolean model using integer programming and feedback vertex sets. *Int. J. Knowl. Discov. Bioinformatics*, 1:14–31, 2010.

[88] Trinh, C. T., Wlaschin, A., and Srienc, F. Elementary mode analysis: a useful metabolic pathway analysis tool for characterizing cellular metabolism. *Appl. Microbiol. Biotechnol.*, 81(5):813–826, 2009.

[89] Varma, A. and Palsson, B. O. Metabolic flux balancing: basic concepts, scientific and practical use. *Nat. Biotech.*, 12:994–998, 1994.

[90] Wilhelm, T., Behre, J., and Schuster, S. Analysis of structural robustness of metabolic networks. *Syst. Biol.*, 1(1):114–120, 2004.

[91] Wu, M., Li, X. L., Kwoh, C. K., Ng, S. K., and Wong, L. Discovery of protein complexes with core-attachment structures from tandem affinity purification (TAP) data. *J. Comput. Biol.*, 18:1–16, 2011.

[92] Wunderlich, Z. and Mirny, L. A. Using the topology of metabolic networks to predict viability of mutant strains. *Biophys. J.*, 91(6):2304–2311, 2006.

[93] Yamagata, K., Uchida, T., Shoudai, T., and Nakamura, Y. An effective grammar-based compression algorithm for tree structured data. In *Proc. 13th Int. Inductive Logic Programming*, pages 383–400, 2003.

[94] Yamamoto, N., Nakahigashi, K., Nakamichi, T., Yoshino, M., Takai, Y., Touda, Y., Furubayashi, A., Kinjyo, S., Dose, H., Hasegawa, M., Datsenko, K. A., Nakayashiki, T., Tomita, M., Wanner, B. L., and Mori, H. Update on the keio collection of escherichia coli single-gene deletion mutants. *Mol. Syst. Biol.*, 5(335):1–3, 2009.

[95] Yook, S. H., Oltvai, Z. N., and Barabási, A. L. Functional and topological characterization of protein interaction networks. *Proteomics*, 4:928–942, 2004.

[96] Zhao, Y., Hayashida, M.and Nacher, J., Nagamochi, H., and Akutsu, T. Protein complex prediction via improved verification methods using constrained domain-domain matching. *Int. J. Bioinformatics. Res. Appl.*, 8(3-4):210–227, 2012.

[97] Zhao, Y., Hayashida, M., and Akutsu, T. Integer programming-based method for grammar-based tree compression and its application to pattern extraction of glycan tree structures. *BMC Bioinformatics*, 11:S4, 2010.

[98] Zhao, Y., Hayashida, M., Jindalertudomdee, J., Nagamochi, H., and Akutsu, T. Breadth-first search approach to enumeration of tree-like chemical compounds. *J. Bioinformatics. Comput. Biol.*, 11(6):1143007, 2013.

[99] Zhao, Y., Tamura, T., Akutsu, T., and Vert, J. Flux balance impact degree: a new definition of impact degree to properly treat reversible reactions in metabolic networks. *Bioinformatics*, 29(17):2178–2185, 2013.

List of Publications

Journal Paper

1. Zhao, Y., Hayashida, M., Jindalertudomdee, J., Nagamochi, H. and Akutsu, T. Breadth-first search approach to enumeration of tree-like chemical compounds, *Journal of Bioinformatics and Computational Biology*, **11**(6), 1143007 (2013).

2. Zhao, Y., Tamura, T., Akutsu, T. and Vert, JP. Flux balance impact degree: a new definition of impact degree to properly treat reversible reactions in metabolic networks, *Bioinformatics*, **29**(17), 2178-2185 (2013).

3. Akutsu, T., Zhao, Y., Hayashida, M. and Tamura, T. Interger programming-based approach to attractor detection and control of boolean networks, *IEICE Transactions on Informatics and Systems*, **E95-D**(12), 2960-2970 (2012).

4. Zhao, Y., Hayashida, M., Nacher, J., Nagamochi, H. and Akutsu, T. Protein complex prediction via improved verification methods using constrained domain-domain matching, *International Journal of Bioinformatics Research and Applications*, **8**(3-4), 210-227 (2012).

5. Zhao, Y., Hayashida, M., and Akutsu, T. Integer programming-based method for grammar-based tree compression and its application to pattern extraction of glycan tree structures, *BMC Bioinformatics*, **11**(Suppl 11): S4 (2010).

Conference Paper

1. Zhao, Y., Tamura, T., Hayashida, M., and Akutsu, T. A dynamic programming algorithm to predict synthesis processes of tree-structured compounds with graph grammar, In *Proceedings of the* 10^{th} *International Workshop on Bioinformatics and Systems Biology*, 218-229 (2010).